BU Summer Challenge 2

Computer Science

Instructor: Karan Vombatkere
July 2023

Course Overview

1. **Computer Science basics** - What is Computer Science? Software vs. Hardware, What is an Algorithm? What tasks can be automated? Data representation, Abstractions
2. **Python programming** - Data types, variables, conditionals, loops, functions, classes, recursion
3. **Exposure to breadth of CS topics** - Artificial Intelligence, Algorithmic theory, Search and optimization, Natural Language Processing,

Reference Book: Algorithms to Live By (Brian Christian and Tom Griffiths)

Group project - final presentations on July 21

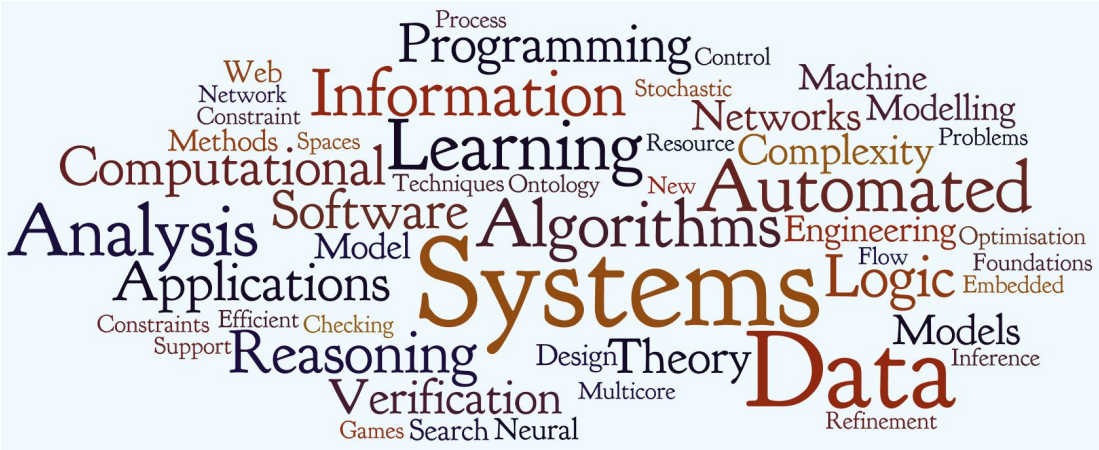
No assignments or grades!

Introductions + Group Activity

- **Introduce yourself (name, class, hometown)**
- **Why are you interested in this course?**
- **Application of Computer Science that interests you**

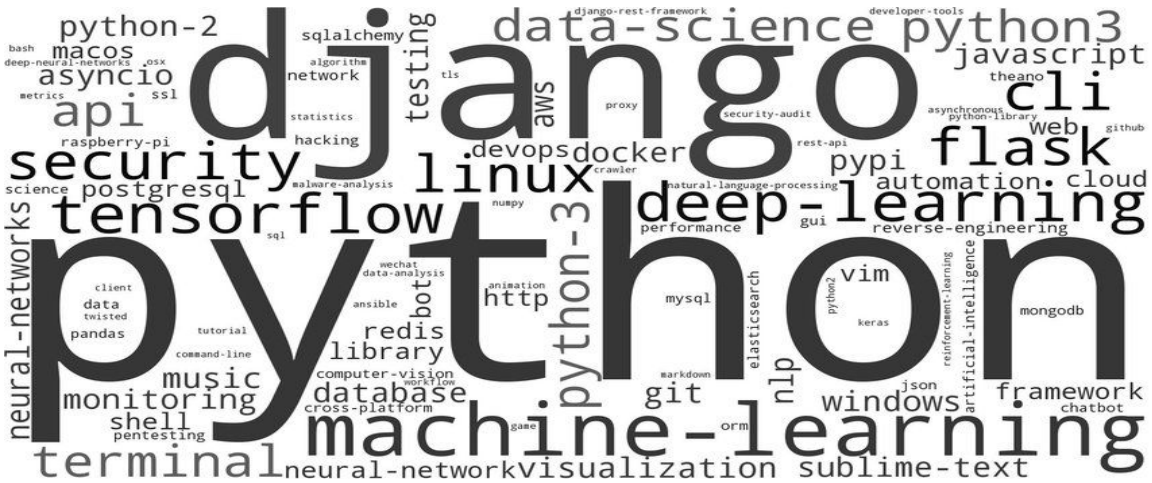
- **Sorting Game**
  - Split into two groups and stand in a line.
  - You are only allowed to communicate with the person to your immediate left/right.
  - The goal is to sort yourselves in alphabetical order by first name.

What is Computer Science?



- Areas of Computer Science
- Software vs. Hardware
- Data and Representations
- Computation and algorithms

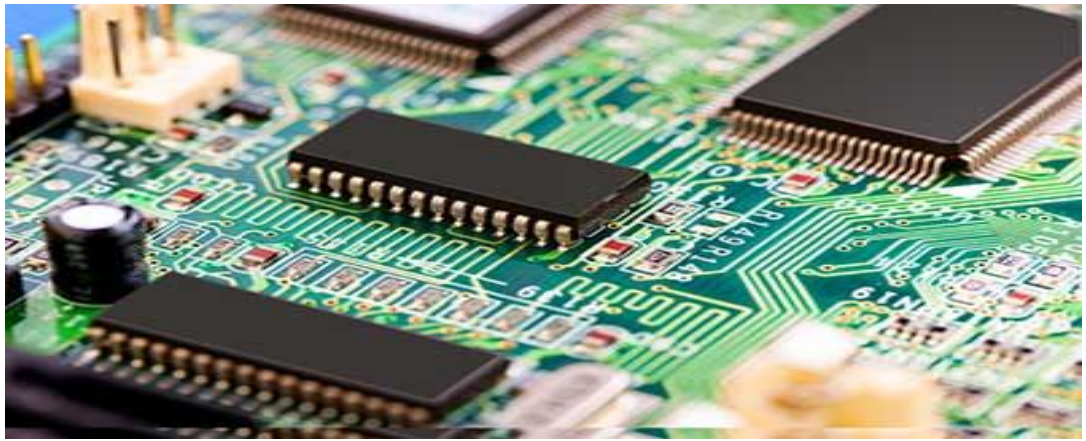
What is Computer Science?



- Programming Languages
- Applications of Computer Science

Computer Hardware

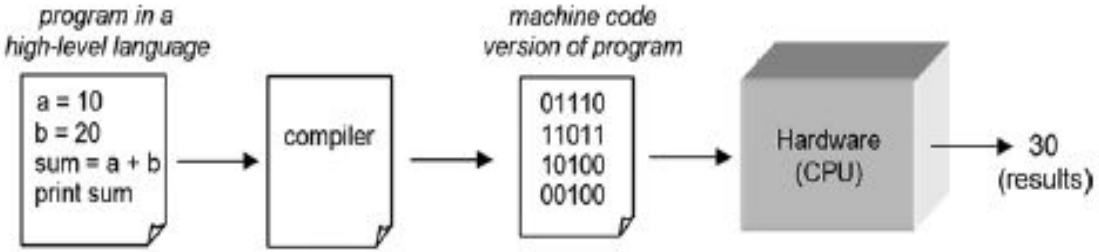
- Central Processing Unit and Main Memory
- Peripheral Components: Keyboard, Monitor, Printer
- Transistors and Integrated Circuits
    - Simple electronic ON/OFF switches

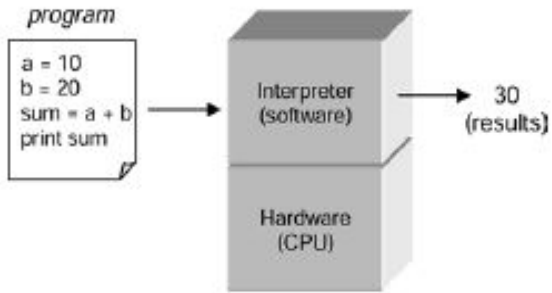


Computer Software

- **Operating system:** Machine code, CPU
- **Low-level vs. High-level language**
- **Compilers and Interpreters**
  - *A compiler is a translator program that translates programs directly into machine code to be executed by the CPU. An interpreter executes program instructions in place of ("running on top of") the CPU.*

- **Syntax:** Set of characters and the acceptable sequences of those characters
- **Semantics:** Meaning associated with characters





Representing Data - Digitization

- **Decimal system - Base 10**
  - Regular numbers with 9 digits
- **Binary System - Base 2**
  - ON/OFF switches - 0 or 1
  - **Bit**: Binary Digit
  - **Byte**: Group of bits, usually consisting of 8 bits

Binary to Decimal

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1024	512	256	128	64	32	16	8	4	2	1
				1	0	1	1	0	0	1

$1011001_2 = 1 \times 64 + 1 \times 16 + 1 \times 8 + 1 \times 1 = 89$

Man, Cabbage, Goat, Wolf Problem

Man, Cabbage, Goat, Wolf Problem

A man with a wolf, a goat, and a cabbage must cross a river by boat. The boat can carry only the man and a single item.

If left unattended together, the wolf would eat the goat, or the goat would eat the cabbage. How can they cross the river without anything being eaten?

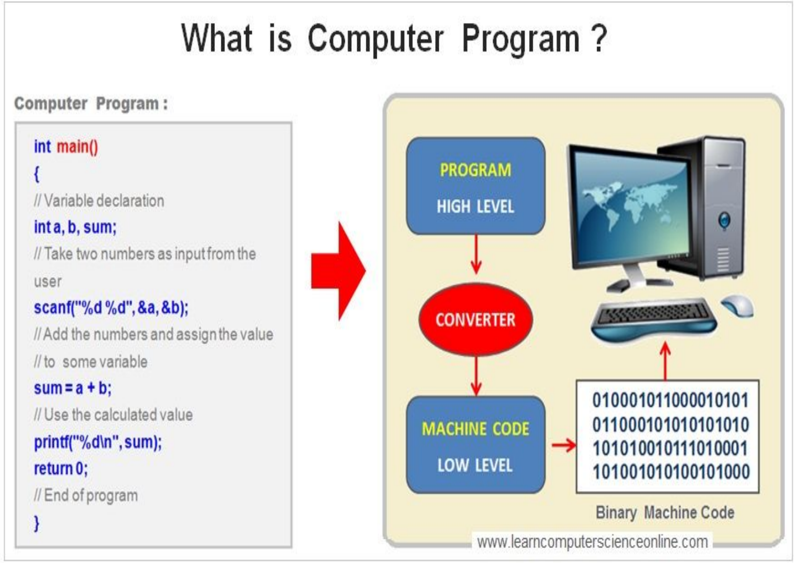
- **What is important and relevant?**
- **Representations and abstraction**
- **What is a solution?**
  - Solution finding algorithm

Computational Problem Solving

- **Problem:** what are we trying to solve? what is the goal?
- **Constraints:** what is allowed?
- **Representation:** capture all the relevant aspects of the problem
- **Algorithm:** technique to solve the problem
- **Time and Computational Resources:** can we solve it quickly?

Python Programming

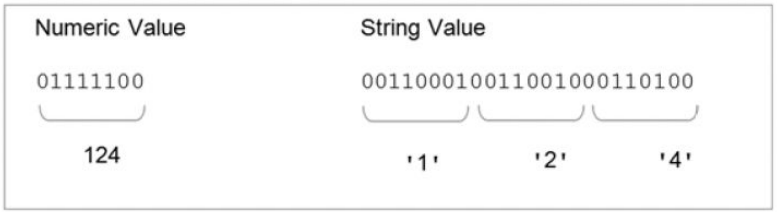
- ★ All computer programs take some **input data, process the data** and then **output results**
- ★ **Variables:** A variable is a name that is assigned to a value.
- ★ **Data Types:** The three types of data most commonly used in Python are **int, float,** and **string**
- ★ **Operators:** Python supports arithmetic and logical operators like addition (+), subtraction (-), multiplication (*) and exponentiation (**).
- ★ **Input and Output:** The **input()** command can be used to obtain user input, and the **print()** function is used to output data.



Python’s Encoding Scheme

- Python uses the UTF-8, an 8-bit encoding scheme. For example, ‘A’ is encoded as **01000001 (65)** and ‘a’ is encoded as **01100001 (97).**
- The **ord** function converts a character to its UTF-8 encoding (**ord(‘A’) = 65)**, and the **chr** function converts a number to its character encoding (**chr(97) = ‘a’)**

Space	00100000	32	A	01000001	65
!	00100001	33	B	01000010	66
"	00100010	34	C	01000011	67
#	00100011	35	.		
.			.		
.			Z	01011010	90
0	00110000	48	a	01100001	97
1	00110001	49	b	01100010	98
2	00110010	50	c	01100011	99
.			.		
.			.		
9	00111001	57	z	01111010	122



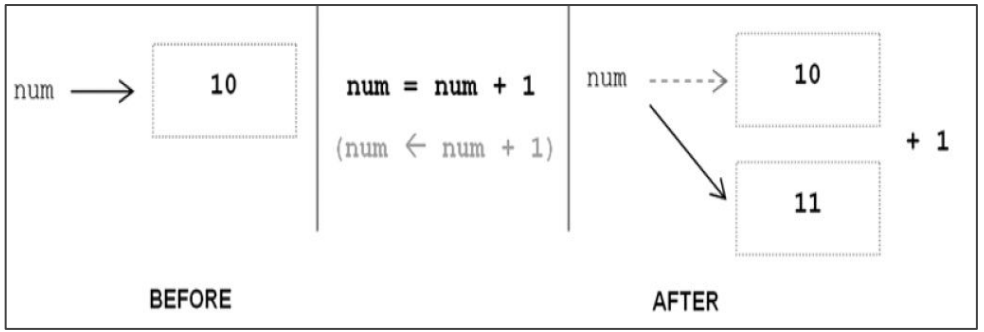
Control Characters

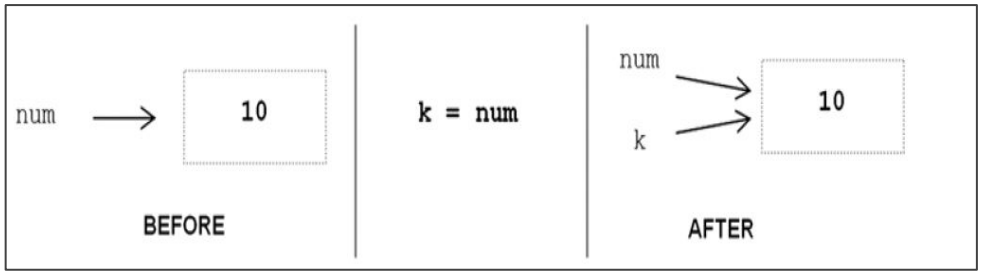
- **S**pecial characters that are not displayed on the screen.
- C*ontrol* the display of output
- Control characters do not have a corresponding keyboard character and represented by a combination of characters called an *escape sequence*.
- The backslash (\) serves as the escape character in Python.
- For example, the escape sequence '\n', represents the *newline control character*, used to begin a new screen line.

Escape Sequence	Meaning
\a	Bell
\b	Backspace
\f	Formfeed
\n	NewLine
\r	Carriage Return
\t	Horizontal Tab
\v	Vertical Tab
\newline	Backslash and NewLine Ignored

Variables

- A variable is a name (identifier) that is associated with a particular value.
- It can be assigned different values during a programs execution.
- Wherever a variable appears in a program (except on the left-hand side of an assignment statement), it is the value associated with the variable that is used , and not the variable's name.
- Variables are assigned values using the **assignment operator (=).**
- Variables can be assigned to the values of other variables.





Identifiers and Keywords

- An **Identifier** is a sequence of one or more characters used to provide a name for a given program element.

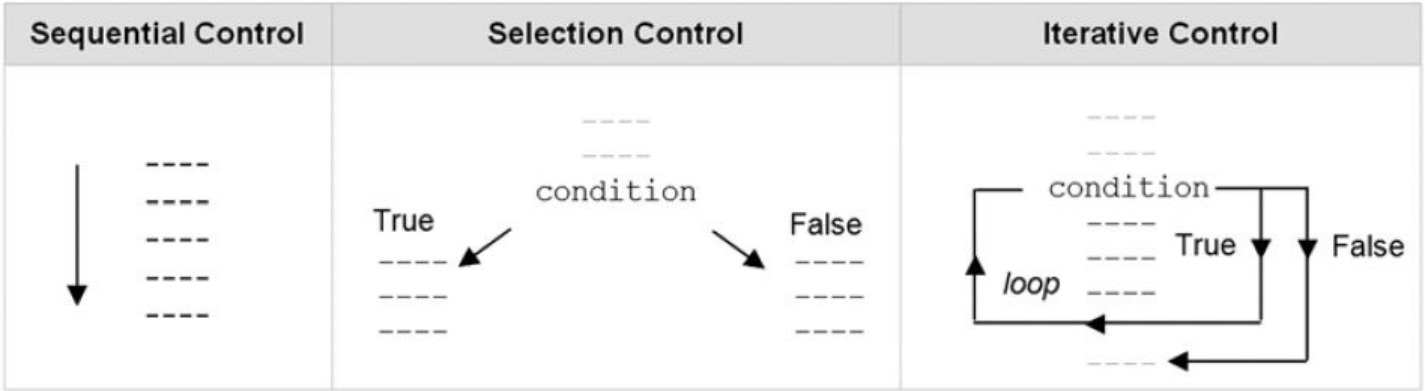
Valid Identifiers	Invalid Identifiers	Reason Invalid
totalSales	'totalSales'	quotes not allowed
totalsales	total sales	spaces not allowed
salesFor2010	2010Sales	cannot begin with a digit
sales_for_2010	_2010Sales	should not begin with an underscore

- A **Keyword** is an identifier that has a predefined meaning, and cannot be used as a regular identifier.

```
and       as        assert    break     class     continue  def
del       elif      else      except    finally   for       from
global    if        import    in        is        lambda    nonlocal
not       or        pass      raise     return    try       while
with      yield     false     none      true
```

Control Structures: Loops and Conditionals

- **Control flow is the order that instructions are executed in a program**. A control statement is a statement that determines the control flow of a set of instructions.
- There are three fundamental forms of control that programming languages provide—sequential control, selection control, and iterative control.



Boolean Expressions (Conditions)

- The Boolean data type contains two Boolean values, denoted as **True** and **False** in Python.
- A Boolean expression is an expression that evaluates to a Boolean value. Boolean expressions are used to denote the conditions for selection and iterative control statements.

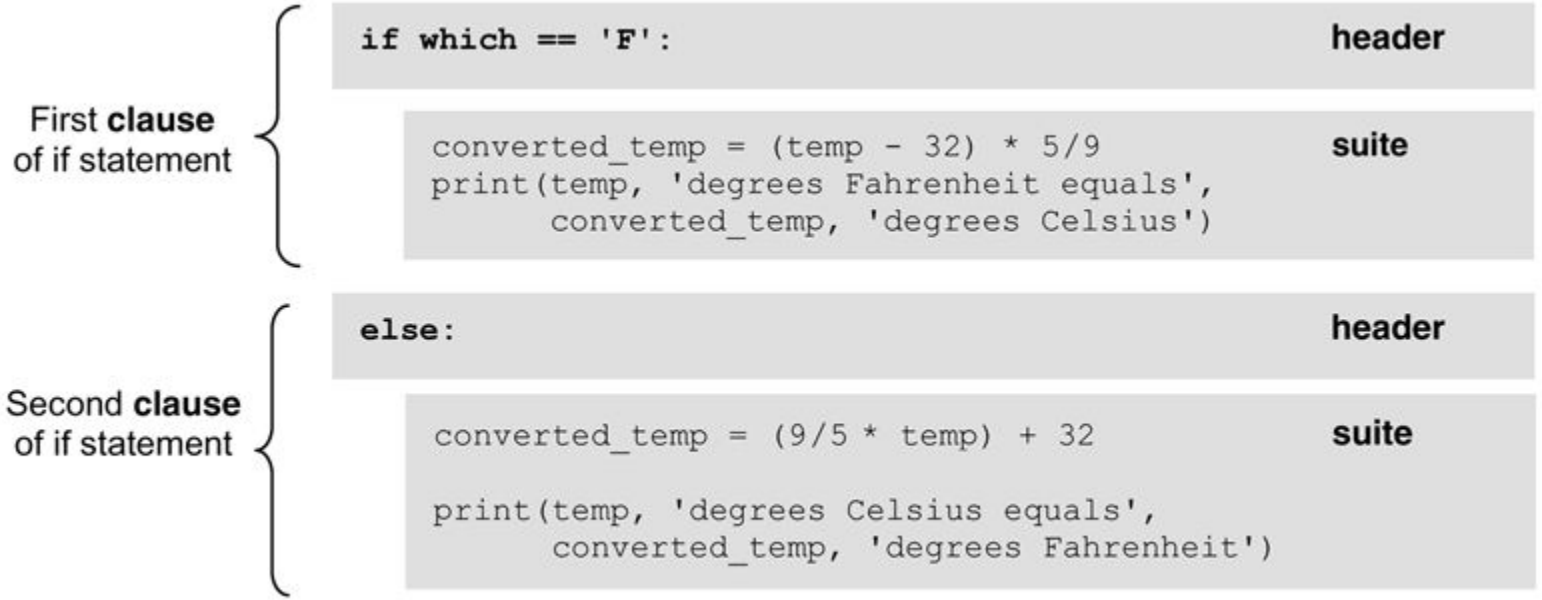
Relational Operators	Example	Result
== equal	10 == 10	True
!= not equal	10 != 10	False
< less than	10 < 20	True
> greater than	'Alan' > 'Brenda'	False
<= less than or equal to	10 <= 10	True
>= greater than or equal to	'A' >= 'D'	False

If-Else Statements

- An if statement is a selection control statement based on the value of a given Boolean expression.
- One fairly unique aspect of Python is that the amount of indentation of each program line is significant - indentation is used to associate and group statements

if statement	Example use	
`if condition:` `    statements` `else:` `    statements`	`if grade >= 70:` `  print('passing grade')` `else:` `  print('failing grade')`	`if grade == 100:` `  print('perfect score!')`

If-Else Statements



If-Else Statements

Valid indentation		Invalid indentation	
(a) ``` if condition: statement statement else: statement statement ```	(b) ``` if condition: statement statement else: statement statement ```	(c) ``` if condition: statement statement else: statement statement ```	(d) ``` if condition: statement statement else: statement statement ```

Nested if statements

```
if condition:
    statements
else:
    if condition:
        statements
    else:
        if condition:
            statements

            etc.
```

Example use

```
if grade >= 90:
    print('Grade of A')
else:
    if grade >= 80:
        print('Grade of B')
    else:
        if grade >= 70:
            print('Grade of C')
        else:
            if grade >= 60:
                print('Grade of D')
            else:
                print('Grade of F')
```

While Loops

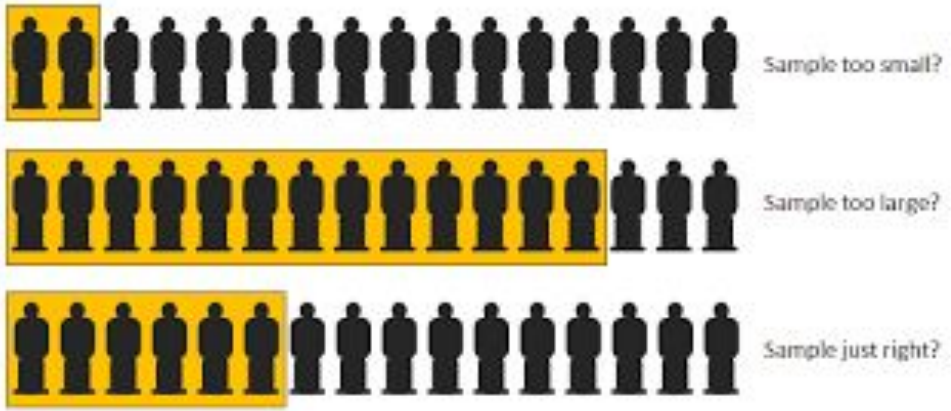
A **while** statement is an iterative control statement that repeatedly executes a set of statements based on a provided Boolean expression (condition). All iterative control needed in a program can be achieved by use of the while statement.

while statement	Example use
`while condition:` `    suite`	`sum = 0` `current = 1` `n = int(input('Enter value: '))` `while current <= n:` `    sum = sum + current` `    current = current + 1`

Optimal Stopping

- **Secretary Problem:** Group Activity

- Finding a Fiance
- Parking a Car
- House Selling



Why is decision making hard?

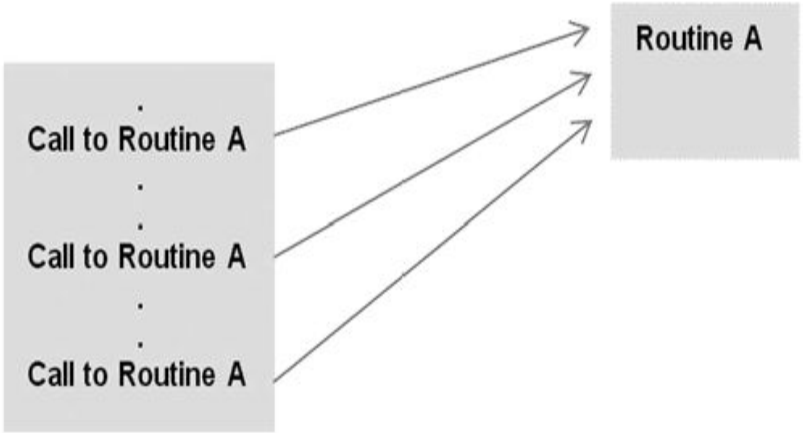
What assumptions do we have in each of these problems?

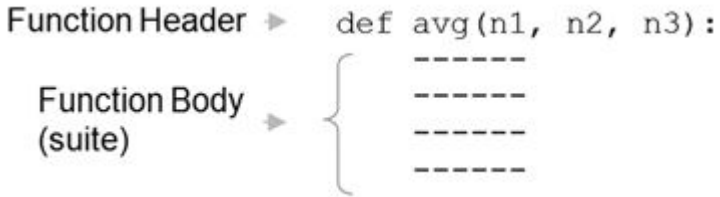
Why do people usually stop earlier?

[Numberphile Video]

Python Functions

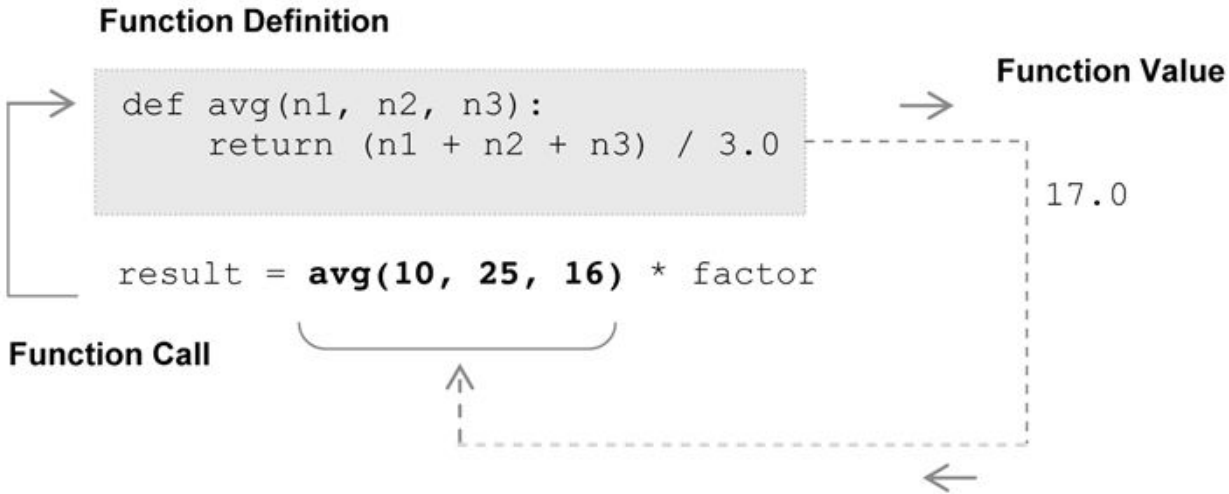
*A **program routine** is a named group of instructions that accomplishes some task. A routine may be invoked (called) as many times as needed in a given program. A **function** is Python's version of a program routine.*





Python Functions

Arguments *are the values passed to functions to be operated on. Parameters are the "placeholder" names for the arguments passed.*



Sorting

- **Why is sorting important? -** Group Activity
- Examples of sorting
- Algorithmic time complexity
  - Efficiency
  - Big-O Notation