4 Horsemen of Coding

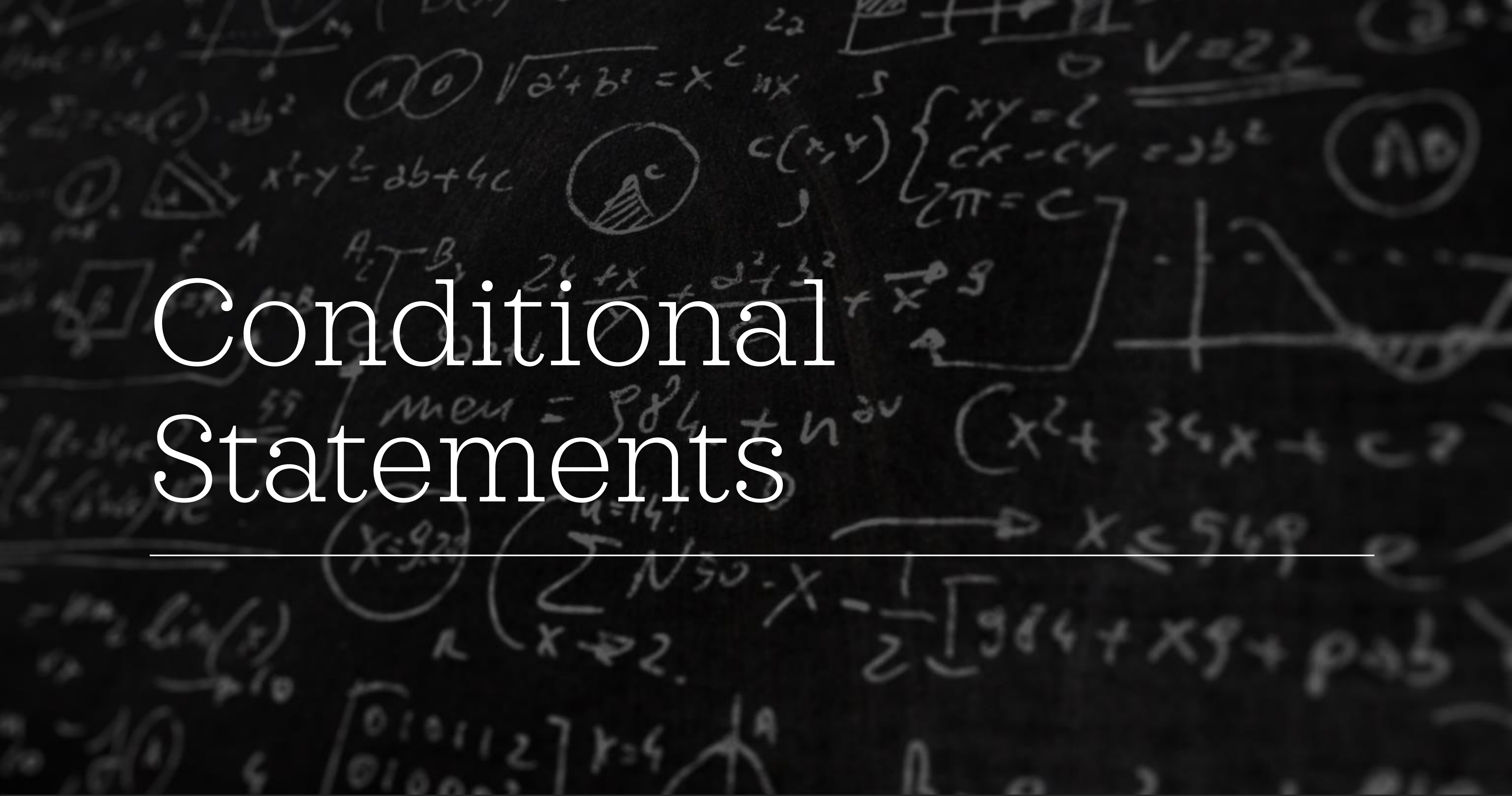
SBS BIOHACKATHON || PYTHON

Scope

1. 4 Horsemen of Coding
2. Conditional Statements
    1. Booleans
    2. Logic Operators
    3. If Else Statement
3. Challenge 2! GC%
4. Functions
5. Challenge 3! Hamming Distance
6. Challenge 4! Exact Match String Search
7. Preview for Next Workshop!

Why & What are the 4 horsemen?

1. For Loops	(Workshop 1)
2. Conditional Statements	(Workshop 2)
3. Functions	(Workshop 2)
4. Reading & Writing from/ to Files	(Workshop 3)



Conditional Statements

Comparison Operators

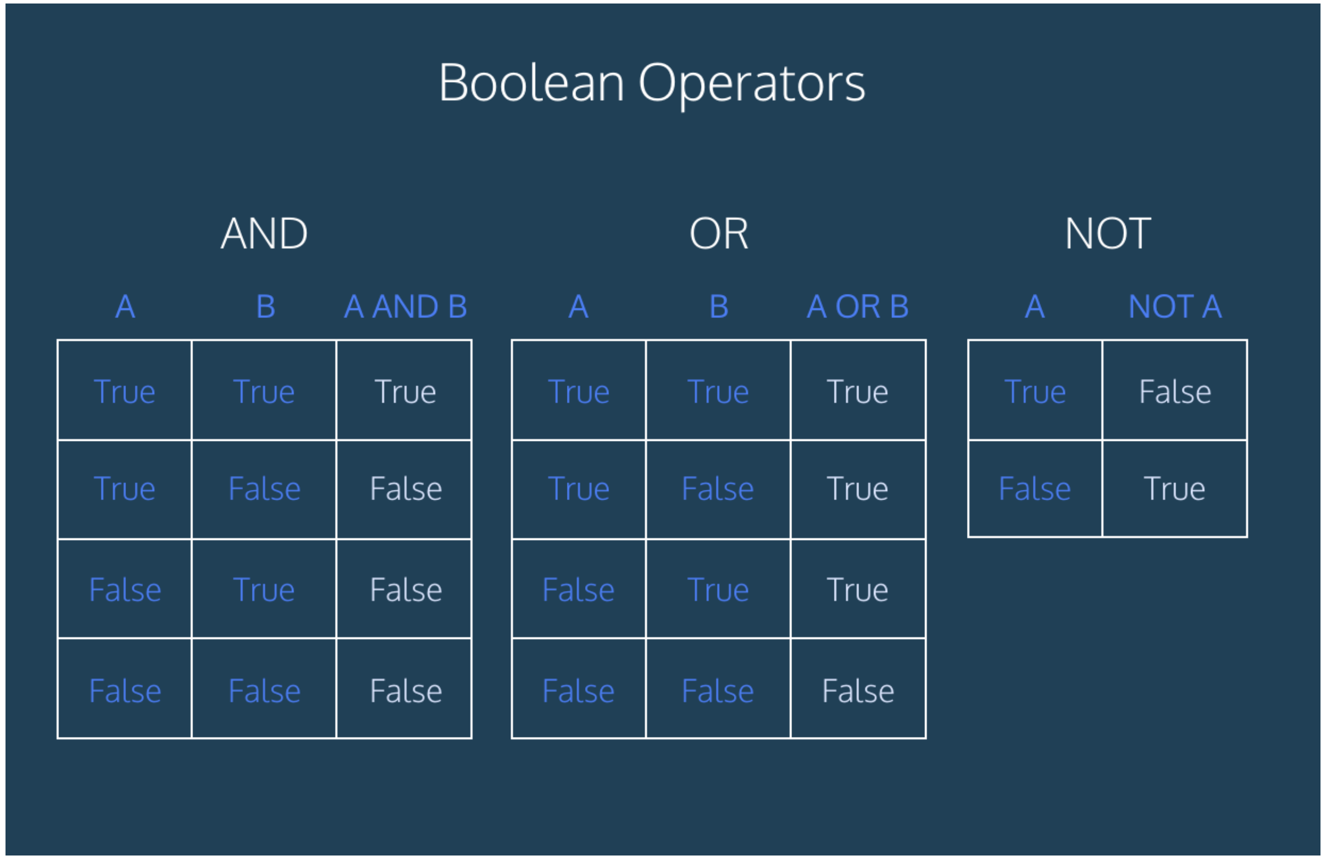
Boolean variables store **True** or **False**

We use comparison operator to compare variables of the same data type

- integer, Character, floats, Strings

Six comparison operators:

1. more than (>)
2. less than (<)
3. equal or more than (>=)
4. equal or less than (<=)
5. equal (==)
6. Not equal (!=)



Boolean Operators

AND

A	B	A AND B
True	True	True
True	False	False
False	True	False
False	False	False

OR

A	B	A OR B
True	True	True
True	False	True
False	True	True
False	False	False

NOT

A	NOT A
True	False
False	True

Logic Operators

```
Hungry      = True
LunchTime   = True
FeelingSad = False

1. If I am Hungry and LunchTime     ; then I get ChickenRice [True A and True B ]
2. If I am Hungry and not LunchTime ; then I get Cookie      [True A and not True B]
3. If I am not Hungry or Sad        ; then I watch Netflix   [not True A or False B]
4. If I am Hungry or Sad            ; then I get food        [True A or False B]
```

Challenge 1

GC %

Your task is to calculate the GC percentage in a DNA String

Challenge 1
GC %

Steps:

1. Store GC count in a variable: GC_count
2. Iterate over the characters in the DNA using For Loop
3. If character is a G or C: GC_count += 1
4. Remember to divide GC_count by len(DNA) * 100

Functions

Challenge 2

Hamming Distance

In information theory, the Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different.

Example:

AGGTGTTCGCTG

AGCTGATCGATG

Hamming Distance = 3

Challenge 2
Hamming Distance

Steps To Take:

1. Optional: Check that the two strings are of equal length
2. Iterate over the length of the strings using range(len(string))
3. Check if the two characters at index *i* are different
4. If different; hamming_distance += 1

Challenge 3
String Search

Your Task is to write a Function to query a motif pattern (len=5) in a DNA sequence and return its index if found.

Challenge 3
String Search

Steps:

1. Iterate using range(len(DNA) – len(motif))
2. Slice the DNA[*i*: *i*+len(motif)] to check if it matches the query
3. If match: return *i*
4. If no match is found: return -1

Preview

		A	**G**	**C**	**T**	**C**	**G**
	0	1	2	3	4	5	6
A	1	0	1	2	3	4	5
G	2	1	0	1	2	3	4
C	3	2	1	0	1	2	3
T	4	3	2	1	0	1	2
A	5	4	3	2	1	1	2
G	6	5	4	3	2	2	1

Challenge 4

We’ve covered Hamming Distance And String Search (Exact Match), how can we combine both concepts to search for string with one or more mismatch (ie. Error)?

Challenge 5

Edit Distance! Measuring dissimilarity between two strings, ie. What’s the least operation for one string 1 to be converted to string 2 in terms of Substitution, Insertion and Deletion