

WEEK 6

# CODE IN PLACE



# AGENDA

01

Check-in

02

Review

03

Coding Challenge/s

VOTE ON WHICH CHALLENGE/S YOU WANT TO DO IN THE CHAT

coding challenge #1

Tracing: Read and analyze code

coding challenge #2

What's that in dog years?: Convert human age to dog years

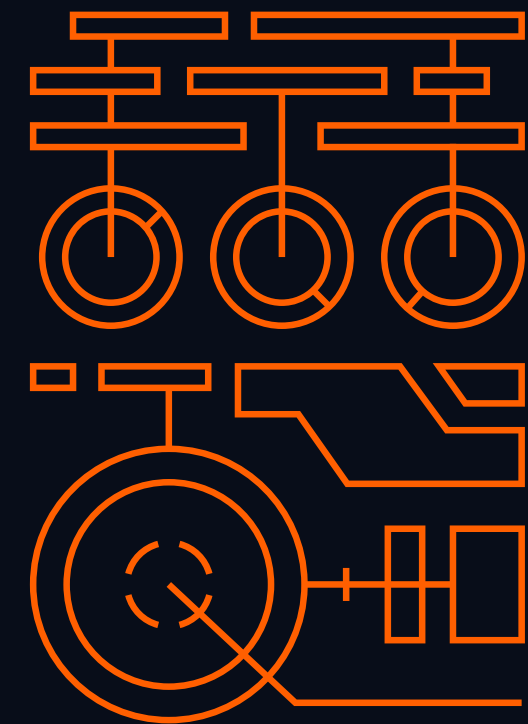
coding challenge #3

Finding factors: Find factors of user's number

coding challenge #4

Rock, paper, scissors: Play rock, paper scissors with computer

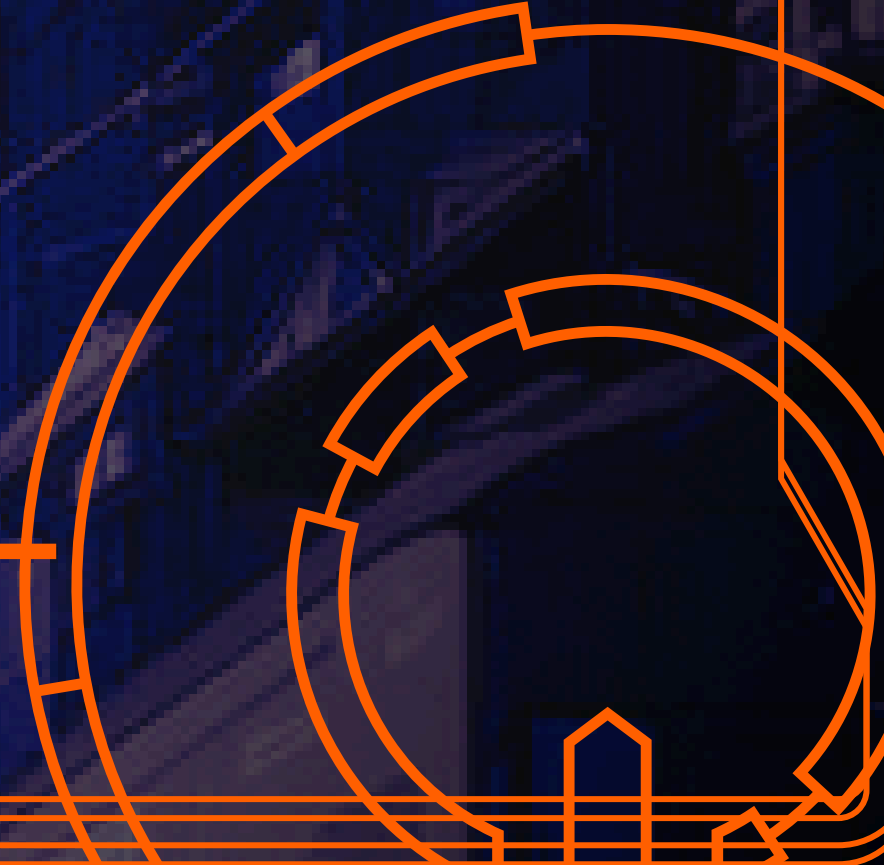
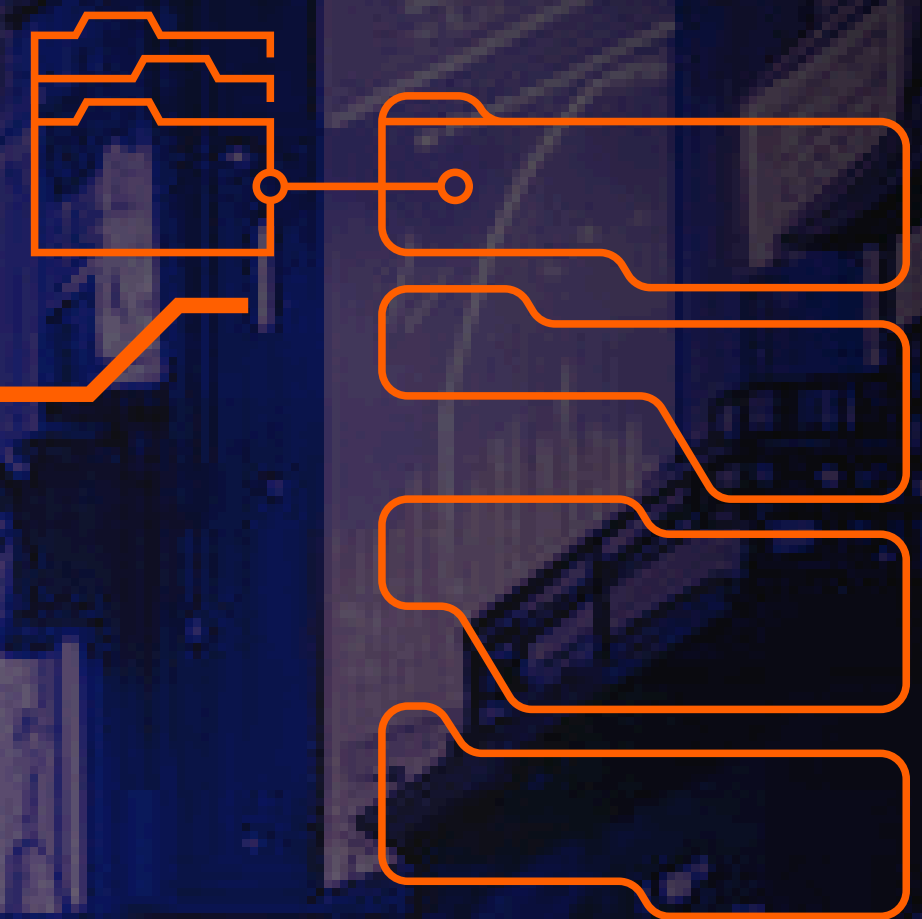
THIS WEEK WE'RE COMBINING THE CODING CHALLENGES WITH THE REVIEW!





# CHECK IN

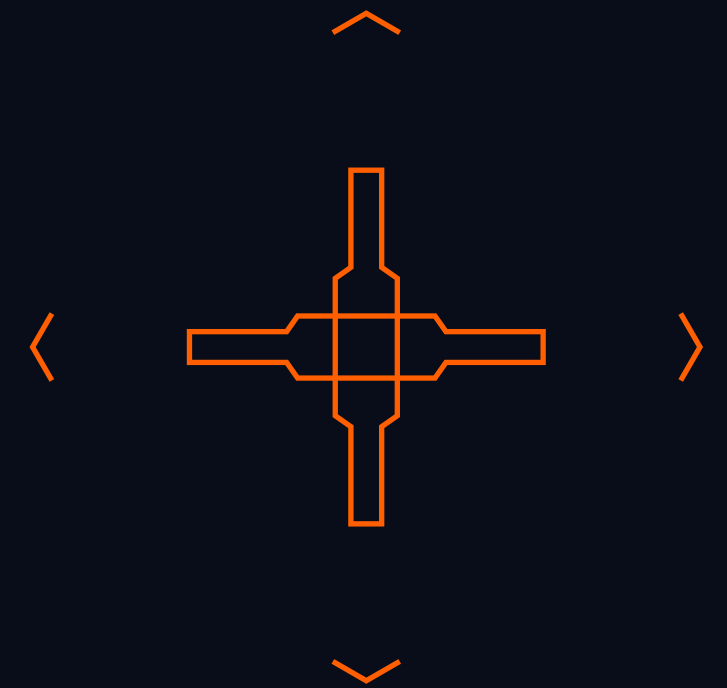
WHAT'S THE FUNNIEST VARIABLE NAME YOU'VE EVER USED?



# REVIEW

(USING CODING  
CHALLENGES)

LET'S START!



## Coding Challenge #1

# TRACING

The following is code for an interactive console program that performs a type of calculation that is probably familiar. Examine the code.

What is the role of the SENTINEL constant?

How do each of the four variables-a, b, x and y-change over time?

Overall, what common task does this program do?

### Variables

Like a suitcase (memory storage) with a tag (name).

PROGRAMMING  
CONCEPT

### Control Flow - While Loop

What is it: Sequence of steps that execute if the condition at the beginning is valid.

What is the condition?  
What type of variable is the condition?

PROGRAMMING  
CONCEPT

```
"""
File: mystery_calculation.py
-----
It's your job to figure out what this program does!
"""

SENTINEL = -1

def main():
    a = int(input("Enter a value for a: "))
    b = int(input("Enter a value for b: "))
    x = int(input("Enter a value for x: "))
    while x != SENTINEL:
        y = a * x + b
        print("Result for x = " + str(x) + " is " + str(y))
        x = int(input("Enter a value for x: "))

if __name__ == "__main__":
    main()
```



# LOOPS

## FOR & WHILE



# for

# while

What are they:

- A For Loop is a loop that repeats a block of code a specific number of times.
- It's commonly used to iterate over a sequence (like a list or range).

```
main.py
1 def main():
2     for i in range(5): # Loops 5 times, i goes from 0 to 4
3         print(i)      # Output: 0, 1, 2, 3, 4
4
5
6 if __name__ == "__main__":
7     main()
```

- Syntax: for keyword, a loop variable (i), and the range or sequence to loop through.
- Code inside the loop will run for each item in the range or sequence.
- When you know the exact number of times you need to repeat an action.
- Example: Looping through a list, processing each item in a collection, or repeating a fixed action.

How to use:

- A While Loop repeats a block of code as long as a certain condition is True.
- It keeps going until the condition becomes False.

```
main.py
1 def main():
2     count = 0
3     while count < 5: # Loops as long as count is less than 5
4         print(count) # Output: 0, 1, 2, 3, 4
5         count += 1  # Increments count
6
7 if __name__ == "__main__":
8     main()
```

- Syntax: while keyword followed by a condition.
- Code inside the loop will keep running until the condition is no longer true.
- When you don't know the exact number of repetitions needed.
- Example: Waiting for user input, running a process until a certain state is reached, or looping until a condition is met.

When to use:



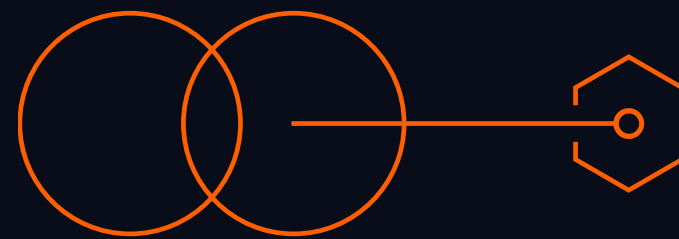
# CONDITIONS

IF, ELIF,  
& ELSE

What are they:

How to use:

When to use:



## if

- Checks a condition. If it's True, the code under it runs.

## elif

- Checks a condition. If it's True, the code under it runs.

## else

- Catches all remaining cases if no previous conditions were True.

```
main.py
1 score = 85
2 if score >= 90:
3     print("A")
4 elif score >= 80:
5     print("B")
6 else:
7     print("C")
8
9 if __name__ == "__main__":
10     main()
```

- Syntax: for keyword, a loop variable (i), and the range or sequence to loop through.
- Code inside the loop will run for each item in the range or sequence.

- Use when you need to make a decision between multiple conditions.
- Example: Grading, choosing options based on inputs, or checking various states.





## Coding Challenge #2

# WHAT'S THAT IN DOG YEARS?

Everyone knows that our furry friends age at a different rate to humans.

Write a program that takes asks the user for a human age (expressed as a whole number) and prints the equivalent dog age using the fact that there are seven dog years per human year.

Consider defining a constant `DOG_YEARS_PER_HUMAN_YEAR`.

You can assume the user types in an integer age, but not necessarily that the integer is positive.

If it isn't, print an error message.

Your program should continuously ask the user for human ages until the user types `0` at which the program should end.

**PROGRAMMING CONCEPT**  
*Constant Variables*  
Use snake case with caps  
What could a `SENTINAL` help with?

**PROGRAMMING CONCEPT**  
*Control Flow - While/If*  
What is it: Used for control flow >> Continuous loop until `0` is input  
How to use it: `while` user-input [operator] [Condition to enter sequence]:

Here is an example of what one run of your program should look like:

```
Enter an age in human years: -12
```

```
Sorry, please enter a positive number or 0 to exit
```

```
Enter an age in human years: 13
```

```
The age in dog years is 91
```

```
Enter an age in human years: 0
```




# BOOLEAN COMPARISON, OPERATORS, PRECEDENCE

What are they:

How to use:

When to use:



## boolean statement

- A statement that is True or False.

## comparison operators (==, !=, <, >, <=, >=)

- Compare values and return True or False.

## logical operators (not, and, or)

- Combine multiple conditions.

## precedence

- Order of operations - Arithmetic > Comparison > Not > And/Or.
- Left to right

```
main.py
1 x = 10
2 y = 20
3 if (x < y) and (y > 15):
4     print("Condition met!")
5
6 if __name__ == "__main__":
7     main()
```

- Use comparison operators within conditions to check values.
- Use not, and, and or to build complex conditions.
- Precedence ensures arithmetic is evaluated first, followed by comparisons, then not, and, and or in that order.

- Use when you need to test conditions or combine multiple checks.
- Examples: Checking if a number is in a range, combining multiple criteria, or flipping a condition's result.



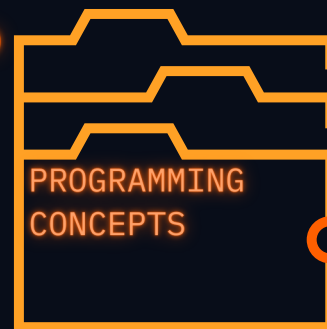
## Coding Challenge #3

# FINDING FACTORS

Implement a program that asks the user to enter an integer, then print out all the factors of the given number, one by one.

Your function should check that the entered number is greater than 0.

The program should keep asking for numbers until 0 is entered.



### While/If

Continuous program execution until condition is invalid.  
Repeated checks

### Logical Expressions

Combined True/False statements

### Comparison Operators

Use for input validation.

`==` `!=` `<`  
`>` `<=` `>=`

### For Loops

`range (start, stop)`

Printing multiple factors

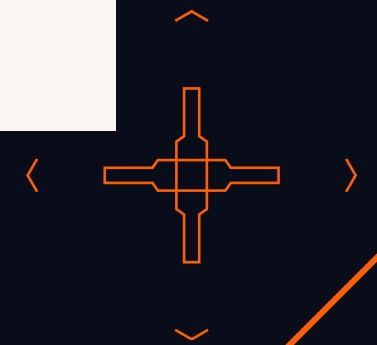
### Factors

A number that divides another number exactly, without leaving a remainder.



Here is an example of what one run of your program should look like:

```
Your number: -10
Please input a positive number
Your number: 42
1
2
3
6
7
14
21
42
Your number: 53
1
53
Your number: 0
```





## Coding Challenge #4

# ROCK, PAPER, SCISSORS

In 1997, a computer named Deep Blue beat world chess champion Gary Kasparov at a game of chess. In 2024, IBM has finally gained the confidence to expand its repertoire of human vs. computer games and enlisted you to write a program that allows a human player and computer to spar over a game of Rock Paper Scissors.

Each game consists of 5 rounds of Rock, Paper, Scissors.

Each round consists of you--the user--choosing whether to play Rock, Paper or Scissors, and the computer doing the same. In this game, the user will type 1 if they wish to play Rock, 2 if they wish to play Paper and 3 if they wish to play Scissors.

In each round, you should follow the following steps:

1. Prompt the user to enter their move (you can assume they type '1', '2' or '3'.)
2. Randomly choose either rock, paper, or scissors as the computer's move using the random module (remember, calling `random.randint(a, b)` will give you back an integer between a and b, inclusive).
3. Determine who wins the round. As a reminder, rock beats scissors, scissors beats paper, and paper, somewhat inexplicably, beats rock.
4. Print a message reporting whether the human player won or lost and which moves each player made.

At the end of the program, you should print a message saying how many rounds you won.

### PROGRAMMING CONCEPT

#### Logical Expressions

What is it: Combinations of booleans with logical operators

How to use it: If [Variable] [Operator]:  
Embed If statements in other If statements or a While loop

### PROGRAMMING CONCEPT

#### Comparison Operators

Use to make value comparisons

==    !=    <  
>    <=    >=

Here is an example of what one run of your program should look like:

```
Your move (1, 2, or 3): 1
It's a tie!
Your move (1, 2, or 3): 3
You Win! Scissors cuts paper
Your move (1, 2, or 3): 2
You Lose! Scissors cuts paper
Your move (1, 2, or 3): 2
It's a tie!
Your move (1, 2, or 3): 1
You Win! Rock crushes scissors
You won 2 rounds!
```