Welcome to the CoGrammar Skills Bootcamp: Logical Programming - Operators

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.

CoGrammar

Cyber Security Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

Cyber Security Session Housekeeping cont.

- For all **non-academic questions**, please submit a query: www.hyperiondev.com/support

- We would love your **feedback** on lectures: Feedback on Lectures

- Find all the lecture **content** in you Lecture Backpack on GitHub.

Safeguarding & Welfare

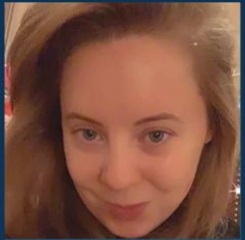
We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:

Ian Wyles
Designated Safeguarding Lead

Simone Botes

Rafiq Manan



Charlotte Witcher

Nurhaan Snyman

Ronald Munodawafa

Tevin Pitts

Scan to report a safeguarding concern

or email the Designated Safeguarding Lead:
Ian Wyles
safeguarding@hyperiondev.com

CoGrammar HyperionDev

Learning Objectives & Outcomes

- Define various types of operators (Mathematical, Relational, Logical).
- Use operators in real-world programming tasks.
- Apply operators to solve simple to complex problems.
- Explain the order of precedence in operations.

CoGrammar

SKILLS FOR LIFE

SKILLS BOOTCAMPS

Department for Education

CoGrammar

CyberSecurity

September 2024

CyberSecurity

- Can you think of daily scenarios where you compare two things or perform calculations?

CoGrammar

CyberSecurity

- **Example**:
    - Comparing product prices or checking if a person is old enough to vote.

Introduction to Operators

- **What Are Operators?**
  - Operators are symbols that tell the program to perform specific mathematical, relational, or logical manipulations.
- Operators help create expressions that the computer can evaluate to make decisions or calculations.

CoGrammar

Main Types of Operators

- Mathematical Operators
- Relational Operators
- Logical Operators

CoGrammar

Mathematical (Arithmetic) Operators

- These operators are used to perform basic mathematical calculations like addition, subtraction, multiplication, and division.
- Common Operators:
  - + (Addition)
  - - (Subtraction)
  - * (Multiplication)
  - / (Division)
  - % (Modulo - returns the remainder of division)
  - ** (Exponentiation)
  - // (Floor Division - returns the largest integer smaller than the result)

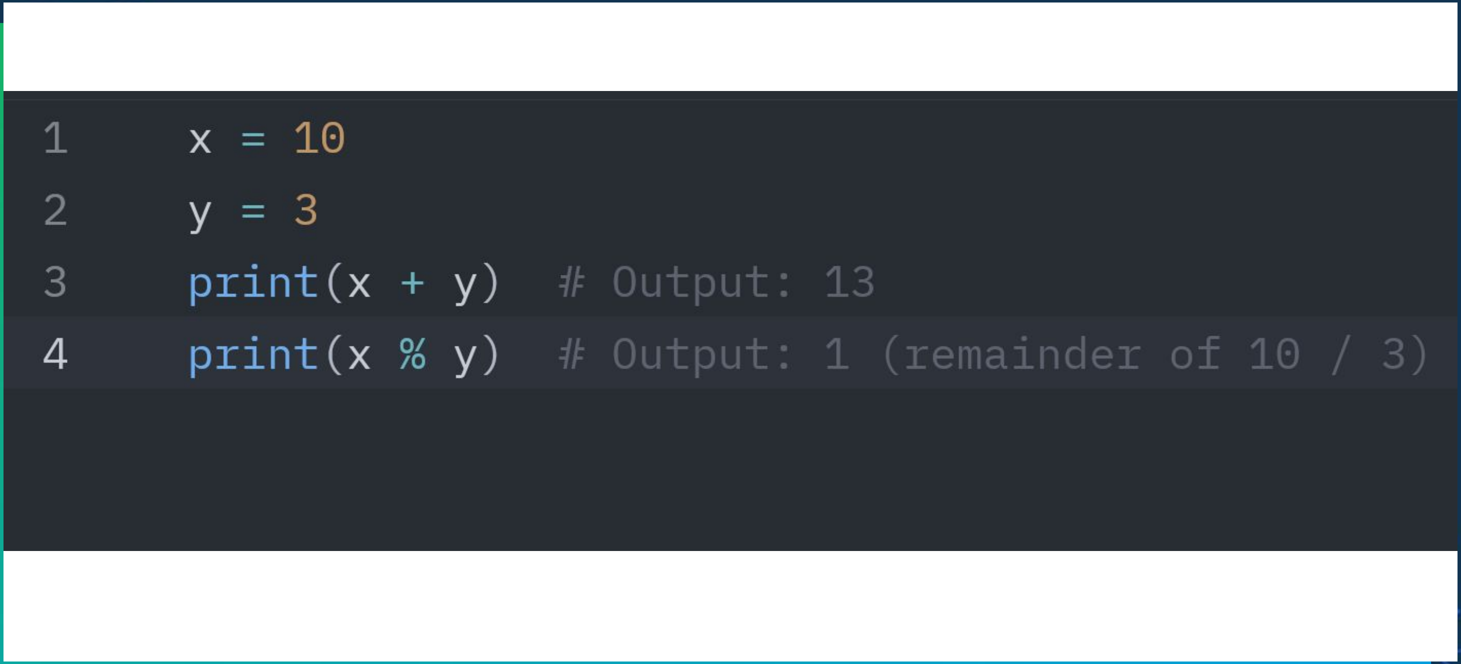
CoGrammar

Mathematical (Arithmetic) Operators

❖ These operators behave similarly to the way we use them in mathematics, but in programming, they follow specific rules about how they're combined and evaluated.

CoGrammar

Example Code:



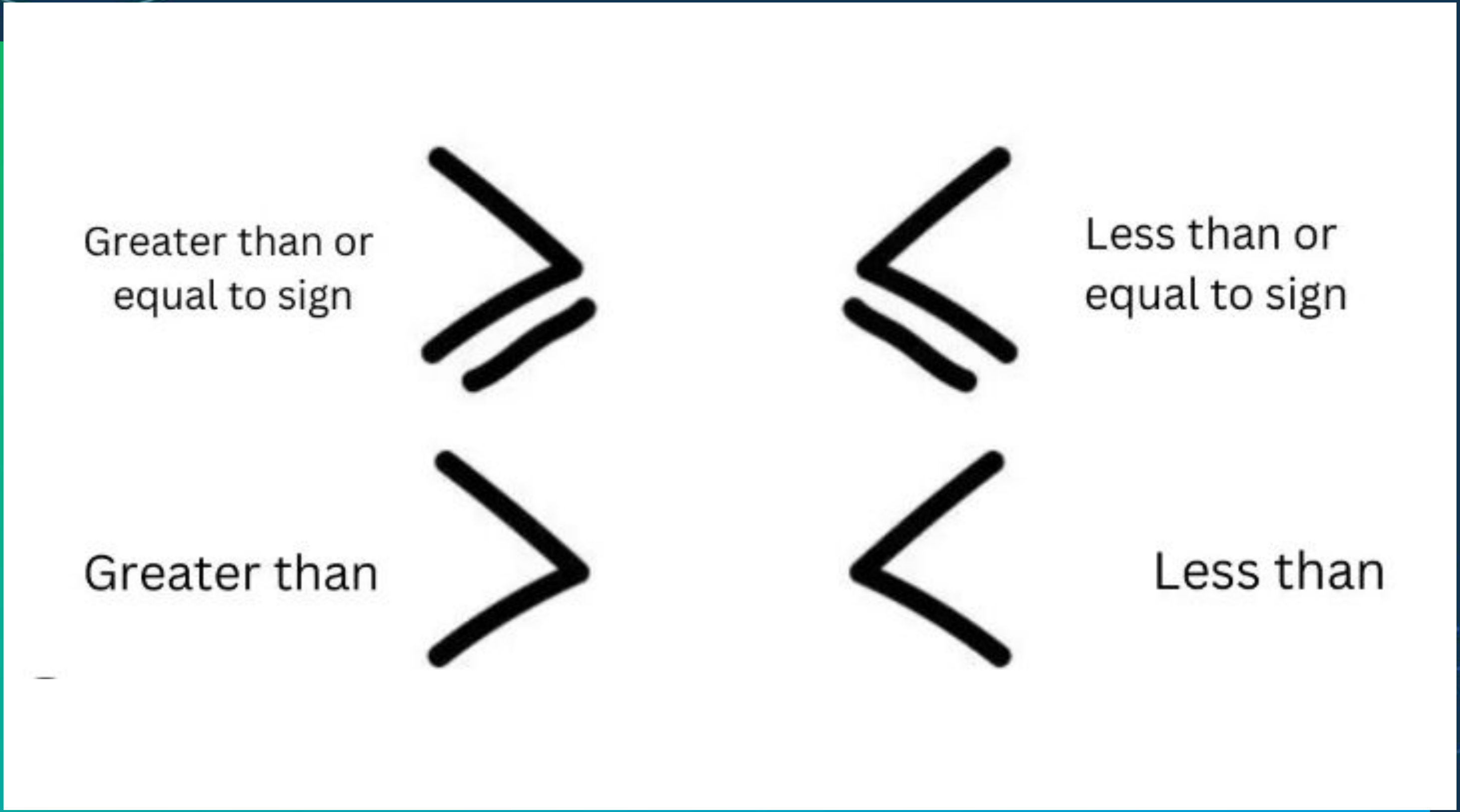
```python
x = 10
y = 3
print(x + y)  # Output: 13
print(x % y)  # Output: 1 (remainder of 10 / 3)
```

CoGrammar

Relational Operators(Comparison)

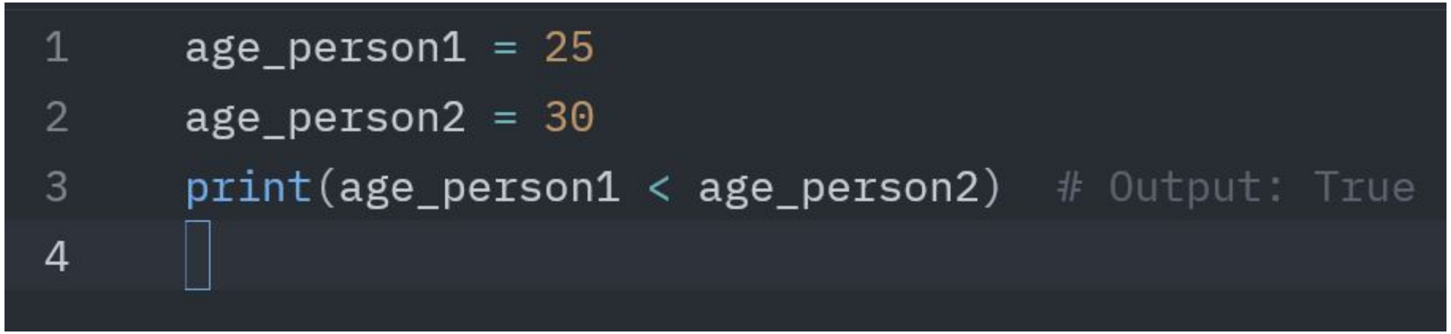
- Relational operators compare two values and return a Boolean (True or False) result based on whether a condition holds.
- Relational expressions are often used in conditional statements
- Common operators:
  - `==` (Equal to)
  - `!=` (Not equal to)
  - `>` (Greater than)
  - `<` (Less than)
  - `>=` (Greater than or equal to)
  - `<=` (Less than or equal to)

CoGrammar


Greater than or equal to sign
Less than or equal to sign
Greater than
Less than

Application Example

- "If we compare the age of two people, we can decide who is older."



```python
age_person1 = 25
age_person2 = 30
print(age_person1 < age_person2)  # Output: True

```

CoGrammar

Let’s take a break
To stretch and relax

CoGrammar

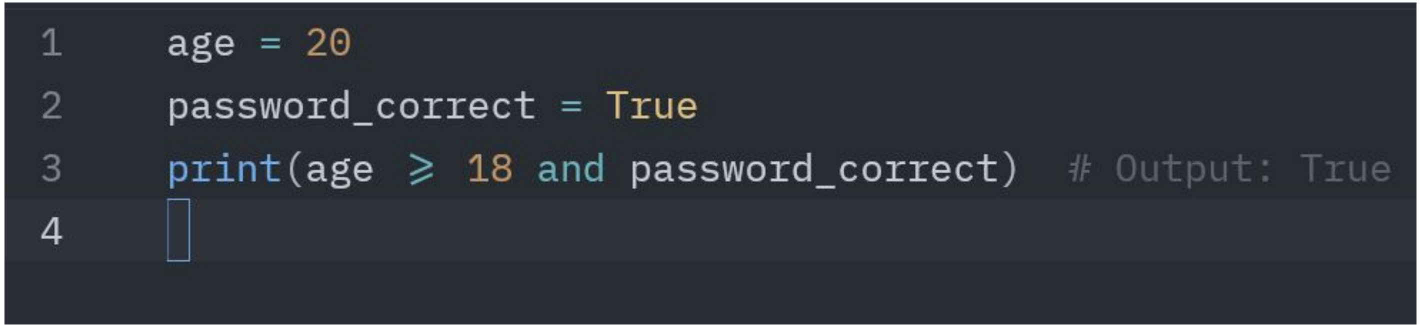
Logical Operators

- Logical operators are used to combine conditional statements and return True or False.
- **Purpose**:
  - These operators are essential in controlling program flow, especially in decision-making processes.
- Common operators:
  - and (True if both operands are true)
  - or (True if at least one operand is true)
  - not (True if the operand is false, and vice versa)
- Logical operators form the backbone of decision-making in complex situations where multiple conditions need to be checked simultaneously.

CoGrammar

Example:

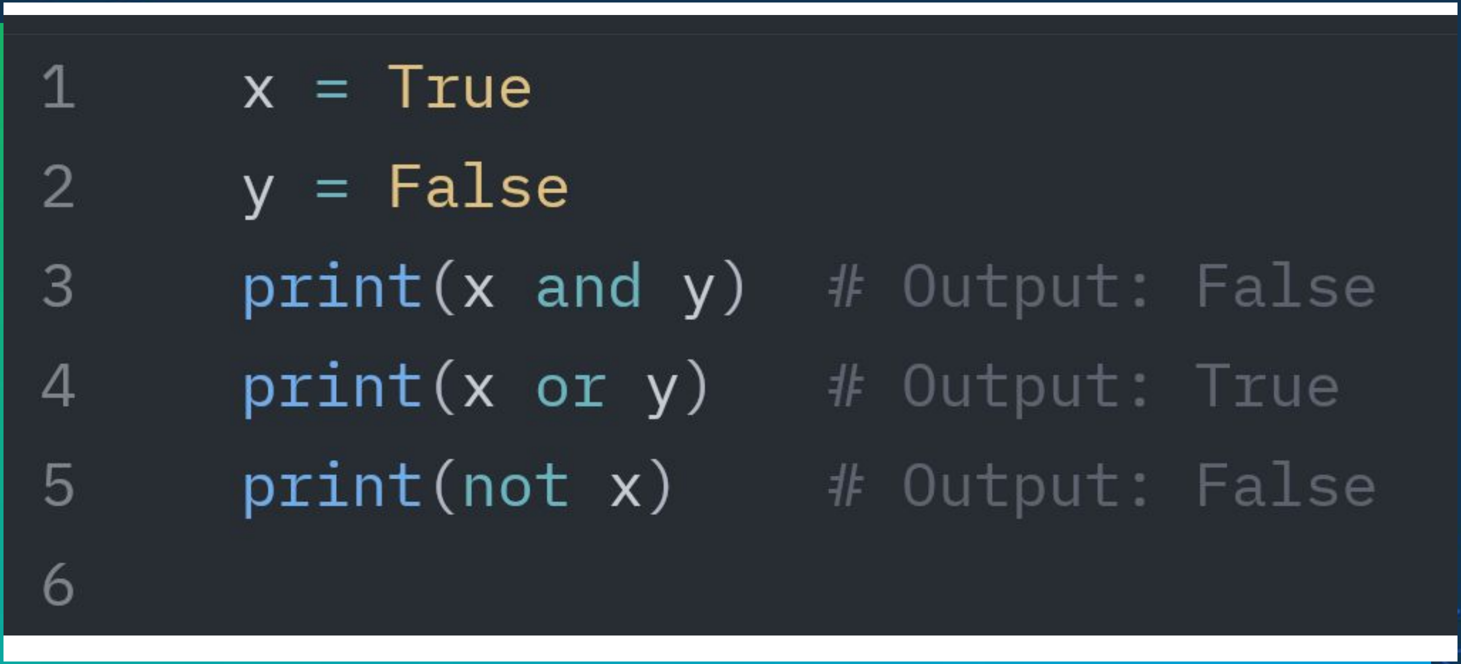
- Consider checking if a user has the right age and the correct password before logging in.



```python
age = 20
password_correct = True
print(age >= 18 and password_correct)  # Output: True

```

Example:



```python
x = True
y = False
print(x and y)  # Output: False
print(x or y)   # Output: True
print(not x)    # Output: False

```

CoGrammar

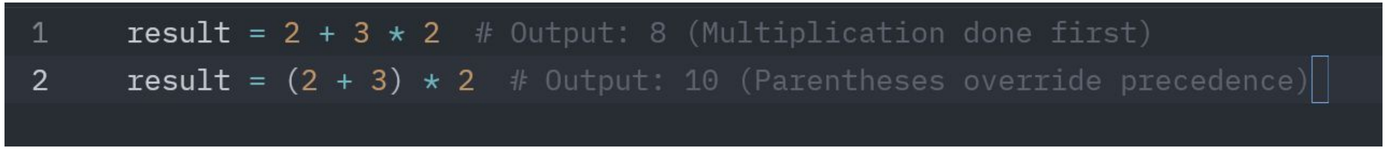
Operator Precedence and Associativity

- **Precedence**:
  - Determines which operator is evaluated first in expressions that have multiple operators.
- **Associativity:**
  - When operators of the same precedence level appear, associativity determines the order of operations.
- **Importance:**
  - Misunderstanding precedence can lead to incorrect program logic, so understanding it ensures accuracy in calculations and comparisons.

CoGrammar

Precedence Table(from highest to lowest)

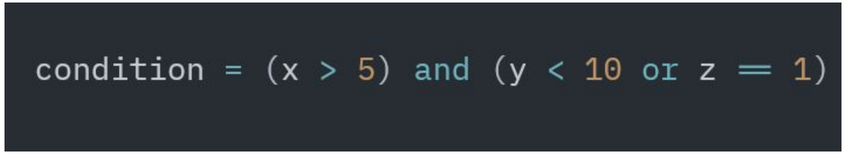
- ** (Exponentiation)
- *, /, //, % (Multiplication, Division, Floor division, Modulus)
- +, - (Addition, Subtraction)
- Relational operators: <, >, <=, >=, ==, !=
- Logical operators: not, and, or



```python
result = 2 + 3 * 2  # Output: 8 (Multiplication done first)
result = (2 + 3) * 2  # Output: 10 (Parentheses override precedence)
```

Operator Precedence in Decision Making

- Operator precedence directly influences the way expressions are evaluated in conditional logic.
- **Practical Implication:**
  - When writing if conditions in a program, knowing how operators are evaluated ensures that the program behaves as expected.
- **Example**:
  - When evaluating complex logical expressions, parentheses are often used to override natural precedence and control the flow.



```
condition = (x > 5) and (y < 10 or z == 1)
```

CoGrammar

Problem Solving with Operators

- Operators are the backbone of problem-solving in programming. They allow us to manipulate data and make decisions.
- Arithmetic and relational operators are often combined in real-world applications to compute values based on conditions. For example, when calculating tax.

CoGrammar

Summary of Operator Types

- **Mathematical Operators:**
  - Perform basic arithmetic.
- **Relational Operators:**
  - Compare values.
- **Logical Operators:**
  - Combine multiple conditions.
- **Key Point:**
  - Operators form the backbone of logical programming, allowing decisions and calculations to be made automatically by the program.

Questions and Answers

CoGrammar

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar