Day 4 25 October 2023

**Introduction to Operators**

Operators in programming are symbols or keywords that perform operations on one or more operands (values or variables). The are used to manipulate data, make comparisons, and control the flow of a program.

Types of operators in programming include:

1. **Arithmetic Operators:**
   * Addition **+**: Adds two values.
   * Subtraction **-**: Subtracts the right operand from the left operand.
   * Multiplication **\***: Multiplies two values.
   * Division **/**: Divides the left operand by the right operand.
   * Modulus **%**: Returns the remainder after division.
   * Exponentiation **\*\***: Raises the left operand to the power of the right operand (in some languages).
2. **Comparison Operators:**
   * Equal to **==**: Checks if two values are equal.
   * Not equal to **!=**: Checks if two values are not equal.
   * Greater than **>**: Checks if the left operand is greater than the right operand.
   * Less than **<**: Checks if the left operand is less than the right operand.
   * Greater than or equal to **>=**: Checks if the left operand is greater than or equal to the right operand.
   * Less than or equal to **<=**: Checks if the left operand is less than or equal to the right operand.
3. **Logical Operators:**
   * Logical AND **&&** (or **and** in some languages): Returns true if both operands are true.
   * Logical OR **||** (or **or** in some languages): Returns true if at least one of the operands is true.
   * Logical NOT **!** (or **not** in some languages): Returns the opposite boolean value of the operand.
4. **Assignment Operators:**
   * Assignment **=**: Assigns the value on the right to the variable on the left.
   * Compound assignment operators (e.g., **+=**, **-=**, **\*=**, **/=**) combine an operation with an assignment. For example, **x += 5** is equivalent to **x = x + 5**.
5. **Bitwise Operators (in languages that support bitwise operations):**
   * Bitwise AND **&**: Performs a bitwise AND operation.
   * Bitwise OR **|**: Performs a bitwise OR operation.
   * Bitwise XOR **^**: Performs a bitwise XOR (exclusive OR) operation.
   * Bitwise NOT **~**: Performs a bitwise NOT operation (inverts the bits).
   * Bitwise left shift **<<**: Shifts the bits of the left operand to the left by the number of positions specified by the right operand.
   * Bitwise right shift **>>**: Shifts the bits of the left operand to the right by the number of positions specified by the right operand.

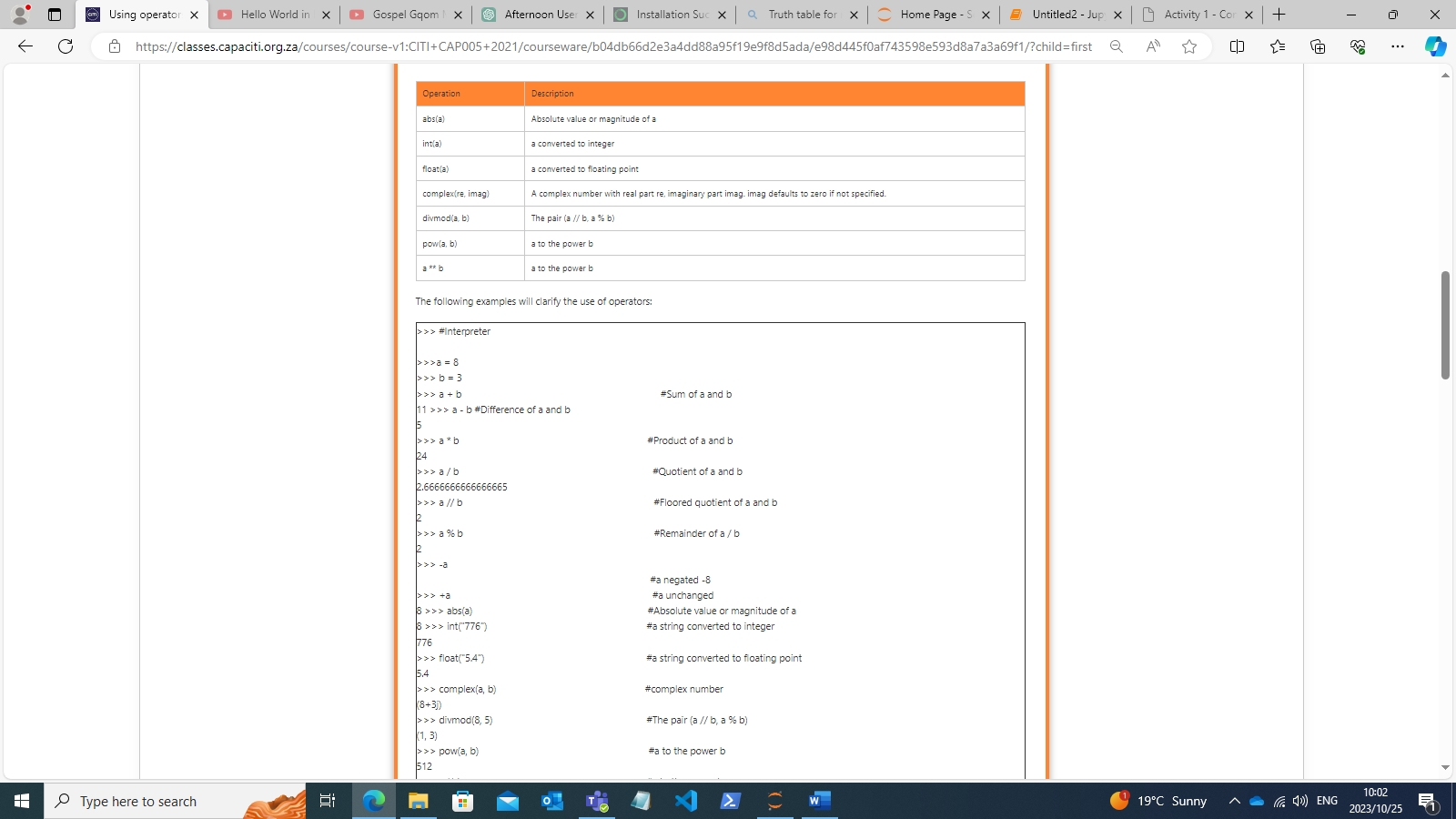
**Using Operators**

Operators require data to operate, and this data is known as operand. This means that when we have **2** and **3** they functions as **operands** for the **+** operator.

This table maps the Mathematical operators:

|  |  |
| --- | --- |
| **Operation** | **Description** |
| a + b | Sum of a and b |
| a - b | Difference between a and b |
| a \* b | Product of a and b |
| a / b | Quotient of a and b |
| a // b | Floored quotient of a and b. Returns an integer value that was implicitly converted from a floating point value.  For example, 8 / 5 = 1.6, when implicitly converted all digits fall away after the dot (.), 8 // 5 = 1. |
| a % b | Remainder of a / b (modular operator) |
| * a | a negated |
| + a | a unchanged |

The table below maps the Mathematical functions:

****