# OPERATORS

DR IAN CORNELIUS

## HELLO

- Learning Objectives:
  1. Understand the different operators built-in to Python
  2. Demonstrate the ability to declare variables and using a variety of operators

## WHAT IS AN OPERATOR?

- An operator is a character that represents an action of some sort
- They are used for performing operations on variables and values (otherwise known as operands)
- Python has a collection of operators built-in:
  - Arithmetic
  - Assignment
  - Comparison
  - Logical
  - Identity
  - Membership

## ARITHMETIC AND ASSIGNMENT OPERATORS (1)

- These operators are used with numeric values to perform mathematical operations

### EQUALS ("=")

- The equal assignment operator is used to assign a value (or another variable) to a variable

```
x = 1
y = 2
z = x
```

```
z = 1
```

## ARITHMETIC AND ASSIGNMENT OPERATORS (2)

### ADDITION (“+ OR +=”)

- When presented with two values or variables will add them together

```
x = 1 + 2
y = 3
z = x + y
z += x
```

```
x = 3
z = 6 (1st Declaration)
z = 9 (2nd Declaration)
```

## ARITHMETIC AND ASSIGNMENT OPERATORS (3)

### SUBTRACTION (“- OR -=”)

- When presented with two values or variables will subtract them from one another

```
x = 1 - 2
y = 3
z = x - y
x -= y
```

x = -1 (1st Declaration)

z = -4

x = -4 (2nd Declaration)

## ARITHMETIC AND ASSIGNMENT OPERATORS (4)

### DIVISION (“/ OR /=”)

- When presented with two values or variable will divide them from one another

```
x = 9 / 3
y = 3
z = x / y
x /= y
z /= x
```

```
x = 3.0 (1st Declaration)
z = 1.0 (1st Declaration)
x = 1.0 (2nd Declaration)
z = 1.0 (2nd Declaration)
```

## ARITHMETIC AND ASSIGNMENT OPERATORS (5)

### FLOOR DIVISION (“// OR //=”)

- When presented with two values or variables it will divide them from one another and return the integer value

```
x = 9.5 // 2
y = 5
z = x // y
z //= y
```

```
x = 4.0
z = 0.0 (1st Declaration)
z = 0.0 (2nd Declaration)
```

## ARITHMETIC AND ASSIGNMENT OPERATORS (6)

### MULTIPLICATION (“* OR *=”)

- When presented with two values or variables it will multiply them together

```
x = 2 * 4
y = 5
z = x * y
x *= y
z *= x
```

x = 8 (1st Declaration)
z = 40 (1st Declaration)
x = 40 (2nd Declaration)
z = 1600 (2nd Declaration)

## ARITHMETIC AND ASSIGNMENT OPERATORS (7)

### EXPONENTIATION (“** OR **=”)

- When presented with two values or variables it will raise the one value/variable to the power of the other

```
x = 2 ** 8
y = 5
z = x ** y
z **= x
```

```
x = 256
z = 1099511627776 (1st Declaration)
z =
35249714121083826571348148398002815464391421343966471060391382605731070276854749365048330296473663862456968155395298373973259049475943113619888833
(2nd Declaration)
```

## ARITHMETIC AND ASSIGNMENT OPERATORS (8)

### MODULUS ("% OR %=")

- When presented with two values or variables it will return the remainder of a division calculation

```
x = 2 % 8
y = 5
z = x % y
x %= y
z %= x
```

```
x = 2 (1st Declaration)
z = 2 (1st Declaration)
x = 2 (2nd Declaration)
z = 0 (2nd Declaration)
```

## COMPARISON OPERATORS (1)

- These operators are used to compare two values together

### SAME AS ("==")

- This operator is used to check if one variable is the same as another

```
x = 3
y = 3
answer1 = (x == y)
y = 5
answer2 = (x == y)
```

```
answer1 = True
answer2 = False
```

## COMPARISON OPERATORS (2)

### NOT EQUAL (“!=”)

- This operator is used to check if one variable is not the same as another

```
x = 3
y = 3
answer1 = (x != y)
y = 5
answer2 = (x != y)
```

```
answer1 = False
answer2 = True
```

## COMPARISON OPERATORS (3)

### GREATER THAN (“>”)

- This operator is used to check if one variable is greater than the other

```
x = 3
y = 3
answer1 = (x > y)
y = 1
answer2 = (x > y)
```

```
answer1 = False
answer2 = True
```

## COMPARISON OPERATORS (4)

### GREATER THAN OR EQUAL TO (“>=”)

- This operator is used to check is one variable is greater than or equal to the other

```
x = 3
y = 3
answer1 = (x >= y)
y = 1
answer2 = (x >= y)
```

```
answer1 = True
answer2 = True
```

## COMPARISON OPERATORS (5)

### LESS THAN (“<”)

- This operator is used to check if one variable is less than the other

```
x = 3
y = 3
answer1 = (x < y)
y = 5
answer2 = (x < y)
```

```
answer1 = False
answer2 = True
```

## COMPARISON OPERATORS (6)

### LESS THAN OR EQUAL TO (“<=”)

- This operator is used to check is one variable is less than or equal to the other

```
x = 3
y = 3
answer1 = (x <= y)
y = 5
answer2 = (x <= y)
x = 10
answer3 = (x <= y)
```

```
answer1 = True
answer2 = True
answer3 = False
```

## LOGICAL OPERATORS (1)

- These operators are used to combine comparison operators together

### AND (“AND”)

- This operator will return `True` if both comparison operators are evaluated to true
  - Otherwise, it will return false if one of the comparison operators is evaluated to false

```
x = 6
```

```
answer1 = (x > 5 and x < 10)
answer2 = (x > 7 and x < 10)
```

```
answer1 = True
answer2 = False
```

## LOGICAL OPERATORS (2)

### OR (“`OR`”)

- This operator will return `True` if one of the comparison operators are evaluated to true

```
x = 6
```

```
answer1 = (x > 5 or x < 4)
```

```
answer1 = True
```

## LOGICAL OPERATORS (3)

### NOT (“NOT”)

- This operator will return the reverse of the evaluated condition
  - If something is `True` it will return as `False` and vice-versa

```
x = 6
```

```
answer1 = (not(x > 5))
```

```
answer1 = False
```

## IDENTITY OPERATORS (1)

- These operators are used to compare objects, but not if they are equal
- They will compare if two objects are the same, with the same memory location

## IS ("IS")

- This operator will return `True` if both variables are the same object

```
x = ["4061CEM", "Programming", "Algorithms"]
y = ["4061CEM", "Programming", "Algorithms"]
z = x
```

```
answer1 = (x is z)
answer2 = (x is y)
answer3 = (x == z)
print(f"answer1 = {answer1}\n")
```

```
answer1 = True
answer2 = False
answer3 = True
```

## IDENTITY OPERATORS (2)

### IS NOT (“IS NOT”)

- This operator will return `True` if both variables are **not** the same object

```
x = ["4061CEM", "Programming", "Algorithms"]
y = ["4061CEM", "Programming", "Algorithms"]
z = x
```

```
answer1 = (x is not z)
answer2 = (x is not y)
answer3 = (x != z)
```

```
answer1 = False
answer2 = True
answer3 = False
```

## MEMBERSHIP OPERATORS (1)

- These operators are used to test if a sequence exists within an object

### IN (“`IN`”)

- This operator will return `True` if a specified value is in a sequence

```
x = ["4061CEM", "Programming", "Algorithms"]
```

```
answer1 = ("Algorithms" in x)
answer2 = ("4061" in "4061CEM")
```

```
answer1 = True
answer2 = True
```

## MEMBERSHIP OPERATORS (2)

### NOT IN ("NOT IN")

- This operator will return True if a specified value is not in the sequence

```
x = ["4061CEM", "Programming", "Algorithms"]
```

```
answer1 = ("Ian Cornelius" not in x)
answer2 = ("Algorithms" not in x)
answer3 = ("4063" not in "4061CEM")
```

```
answer1 = True
answer2 = False
answer3 = True
```

## GOODBYE

- Questions?
  - Post them in the **Community Page** on Aula
- Contact Details:
  - Dr Ian Cornelius, ab6459@coventry.ac.uk