

CONDITIONAL STATEMENTS

DR IAN CORNELIUS

HELLO

- Learning Objectives
 1. Understand what a conditional statement is
 2. Demonstrate the ability to use conditional statements

PREVIOUSLY...

- You were introduced to comparison operators previously
- These operators are used with conditional statements to ensure certain conditions have been met
- Recap on the comparison operators:

Symbol	Explanation
==	The Same
!=	Not the Same
>	Greater Than
>=	Greater Than or Equal To
<	Less Than
<=	Less Than or Equal To !

INTRODUCTION TO CONDITIONAL STATEMENTS

- A basic decision statement which is done using a selection structure
- The decision will be described to the interpreter by a conditional statement
 - whereby a result can only be `True` or `False`
- Python allows the following:
 - `if` statements
 - `if ... else ...` statement
 - `if ... elif ... else` statement
 - Nested `if ... else ...` statements

IF STATEMENTS (1)

- Often referred to as a decision-making statement
- Used to control the flow of execution for statements and to test an expression
 - tests logically whether a condition is **True** or **False**

```
</> if variable == value:  
    ...
```

IF STATEMENTS (2)

```
</> x = 1
```

```
</> if x == 1:  
    print(True)
```

```
▶ True
```

IF ... ELSE STATEMENTS (1)

- Known as an alternative execution, whereby there are two possibilities
 - the condition statement determines which of the two statements gets executed
- The `else` is used as the ultimate result for a test expression
 - this result is only met if all other statements are `False`

```
</> if variable == value:  
    ...  
else:  
    ...
```

IF ... ELSE STATEMENTS (2)

```
</> def is_equal(x):  
    if x == 1:  
        return True  
    else:  
        return False
```

```
</> is_equal(1)  
is_equal("Hello")
```

```
▶ is_equal(1) -> True  
is_equal("Hello") -> False
```


ELSE-IF STATEMENTS (1)

- `elif` is a keyword in Python to replace the `else if` conditions from other languages
- The condition allows for two or more possibilities, known as a **chained conditional**

```
</> if variable > value:  
    ...  
elif variable < value:  
    ...  
else:  
    ...
```

ELSE-IF STATEMENTS (2)

- The first check is to determine whether `x` is equal to `1`
 - `x == 1` the check passes
 - `(True, "x is 1")` is returned

```
</> def is_equal(x, y):  
    if x == 1:  
        return True, "x is 1"  
    elif y == 2:  
        return True, "y is 2"  
    else:  
        return False, "x is not 1 and y is not 2"
```

```
</> is_equal(x=1, y=3)
```

```
▶ is_equal(x=1, y=3) -> (True, 'x is 1')
```

ELSE-IF STATEMENTS (3)

- The first check is to determine whether `x` is equal to `1`
 - `x == 2`, the check fails, move onto the next check
- The second check is actioned, and checks whether `y` is equal to `2`
 - `y == 2`, the check passes
 - `(True, "y is 2")` is returned

```
</> def is_equal(x, y):  
    if x == 1:  
        return True, "x is 1"  
    elif y == 2:  
        return True, "y is 2"  
    else:  
        return False, "x is not 1 and y is not 2"
```

```
</> is_equal(x=2, y=2)
```

```
▶ is_equal(x=2, y=2) -> (True, 'y is 2')
```

ELSE-IF STATEMENTS (3)

- The first check is to determine whether `x` is equal to `1`
 - `x == 2`, the check fails, move onto the next check
- The second check is actioned, and checks whether `y` is equal to `2`
 - `y == 3`, the check fails, move onto the next check
- As both checks have failed, the final case is reached
 - `(False, "x is not 1 and y is not 2")` is returned

```
</> def is_equal(x, y):  
    if x == 1:  
        return True, "x is 1"  
    elif y == 2:  
        return True, "y is 2"  
    else:  
        return False, "x is not 1 and y is not 2"
```

```
</> is_equal(x=2, y=3)
```

```
▶ is_equal(x=2, y=3) -> (False, 'x is not 1 and y is not  
2')
```

NESTED IF ... ELSE STATEMENTS (1)

- `if ... else` statements can be written inside each other
 - this is known as **nesting**

```
</> if variable == value:  
    if variable == value:  
        ...  
    elif variable == value:  
        ...  
    else:  
        ...  
elif variable != value:  
    ...  
else:  
    ...
```

NESTED IF ... ELSE STATEMENTS (2)

- The first check is to determine whether `x` is equal to `1`
 - `x == 1`, the check passes, move onto the next check
- The nested `if` statement is now actioned and a second check is performed on whether `y` is equal to `2`
 - `y == 3`, the check fails, move onto the next check
- The third check is actioned and checks whether `y` is equal to `4`
 - `y == 3`, the check fails, move onto the next check
- The final case is reached
 - `(False, "x is 1 and y is not 2 or 4")` is returned

```
</> def is_equal(x, y):  
    if x == 1:  
        if y == 2:  
            return True, "x is 1 and y is 2"  
        elif y == 4:  
            return True, "x is 1 and y is 4"  
        else:  
            return False, "x is 1 but y is not 2 or 4"  
    else:  
        return "False", "x is not 1"
```

```
</> is_equal(x=1, y=3)
```

```
▶ is_equal(x=1, y=3) -> (False, 'x is 1 but y is not 2 or 4')
```

NESTED IF ... ELSE STATEMENTS (2)

- The first check is to determine whether `x` is equal to `1`
 - `x == 2`, the check fails, move onto the next check
- The nested `if` statement does not execute and the final case is reached
 - `("False", "x is not 1")` is returned

```
</> def is_equal(x, y):  
    if x == 1:  
        if y == 2:  
            return True, "x is 1 and y is 2"  
        elif y == 4:  
            return True, "x is 1 and y is 4"  
        else:  
            return False, "x is 1 but y is not 2 or 4"  
    else:  
        return "False", "x is not 1"
```

```
</> is_equal(x=2, y=3)
```

```
▶ is_equal(x=2, y=3) -> ('False', 'x is not 1')
```

NESTED IF ... ELSE STATEMENTS (3)

- The first check is to determine whether `x` is equal to `1`
 - `x == 1`, the check passes, move onto the next check
- The nested `if` statement is now actioned and a second check is performed on whether `y` is equal to `2`
 - `y == 2`, the check passes
 - `(True, "x is 1 and y is 2")` is returned

```
</> def is_equal(x, y):  
    if x == 1:  
        if y == 2:  
            return True, "x is 1 and y is 2"  
        elif y == 4:  
            return True, "x is 1 and y is 4"  
        else:  
            return False, "x is 1 but y is not 2 or 4"  
    else:  
        return "False", "x is not 1"
```

```
</> is_equal(x=1, y=2)
```

```
▶ is_equal(x=1, y=2) -> (True, 'x is 1 and y is 2')
```


NESTED IF ... ELSE STATEMENTS (4)

- The first check is to determine whether `x` is equal to `1`
 - `x == 1`, the check passes, move onto the next check
- The nested `if` statement is now actioned and a second check is performed on whether `y` is equal to `2`
 - `y == 4`, the check fails, move onto the next check
- The third check is actioned and checks whether `y` is equal to `4`
 - `y == 4`, the check passes
 - `(True, "x is 1 and y is 4")` is returned

```
</> def is_equal(x, y):  
    if x == 1:  
        if y == 2:  
            return True, "x is 1 and y is 2"  
        elif y == 4:  
            return True, "x is 1 and y is 4"  
        else:  
            return False, "x is 1 but y is not 2 or 4"  
    else:  
        return "False", "x is not 1"
```

```
</> is_equal(x=1, y=4)
```

```
▶ is_equal(x=1, y=4) -> (True, 'x is 1 and y is 4')
```

NESTED IF ... ELSE STATEMENTS (5)

- The first check is to determine whether `x` is equal to `1`
 - `x == 1`, the check passes, move onto the next check
- The nested `if` statement is now actioned and a second check is performed on whether `y` is equal to `2`
 - `y == 5`, the check fails, move onto the next check
- The third check is actioned and a check is performed on whether `y` is equal to `4`
 - `y == 5`, the check fails, move onto the next check
- The final case is reached
 - `(False, "x is 1 but y is not 2 or 4")` is returned

```
</> def is_equal(x, y):  
    if x == 1:  
        if y == 2:  
            return True, "x is 1 and y is 2"  
        elif y == 4:  
            return True, "x is 1 and y is 4"  
        else:  
            return False, "x is 1 but y is not 2 or 4"  
    else:  
        return "False", "x is not 1"
```

```
</> is_equal(x=1, y=5)
```

```
▶ is_equal(x=1, y=5) -> (False, 'x is 1 but y is not 2 or  
4')
```

GOODBYE

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