3 - Selection

### General Teacher Notes

#### Key Concepts/Vocab/Misconceptions

This set of exercises will take us through more advanced uses of selection

I’ve put the specific notes in the ‘Teacher Notes’ section for each task. They seem to fit better there.

### 1 - Boolean Operators for More Complex Conditions

#### TEACHER NOTES

In the ‘Intro To’ course we used selection with a single condition to decide whether to execute the if, elif or else instructions.

Here we use **Boolean operators** to allow us to create conditions that check more than one true or false condition.

We are using this technique with selection, but it can be used anywhere you would use conditions, for example in conditional (while) loops.

**Make sure that** pupils are aware that each condition must include the data or variable being checked. Lots write code like this: num1 < 5 and < 10

The operators are:

**and** - checks more than one condition, only returns true if **ALL** conditions are true.

num1 = 9

#returns false, because only one condition (num1 < 10) is true.

num1 < 5 and num1 < 10

#returns true, because both conditions are true.

num1 < 20 and num1 < 10

#returns false, because only two conditions (num1 < 10, num1 < 20 are true)

num1 < 5 and num1 < 10 and num1 < 20

**or** - checks more than one condition, returns true if **AT LEAST ONE** condition is true. Another way of thinking about this is that or only returns false when all conditions are false

num1 = 9

#returns true, because one condition (num1 < 10) is true.

num1 < 5 and num1 < 10

#returns true, because both conditions are true.

num1 < 20 and num1 < 10

#returns false, because all conditions are false

num1 < 5 and num1 < 9 and num1 < 7

**not** - returns the opposite of a condition - if the condition is true it returns false, if it’s false it returns true

num1 = 9

#returns false because the condition is true.

not(num1 < 10)

#returns true because the condition is false.

not(num1 < 5)

#returns true because the condition is false.

not(num1 != 9)

#### Tasks

##### Predict And Run

Task and instructions - <https://repl.it/@MrAColley/231-Multiple-Boolean-Predict-run>

Example solution - <https://repl.it/@MrAColley/231-Multiple-Boolean-Predict-Run-Solution>

##### Investigate And Modify

Task and instructions - <https://repl.it/@MrAColley/232-Multiple-Boolean-Investigate-Modify>

Example solution - <https://repl.it/@MrAColley/232-Multiple-Boolean-Investigate-Modify-Solution>

##### Make

Task and instructions - <https://repl.it/@MrAColley/233-Multiple-Boolean-Make>

Example solution - <https://repl.it/@MrAColley/233-Multiple-Boolean-Make-1>

### 

### 

### 2 - Numbers in a Range

#### TEACHER NOTES

**Validation** is the process of checking whether data is sensible and allowable. It’s a really important part of lots of programs that involve input (for example checking your password).

We can use Boolean operators to check whether a number input is within (or indeed outside) a range.

**Note - we are using selection to output a relevant message. This doesn’t make the program loop to ask for the input again - we will cover that when we look at iteration in the next set of tasks.**

To check if a number is within a range, we use:

Greater than or equal to the lowest value **AND l**ess than or equal to the highest value.

Both conditions have to be true to ensure that the number is within the range.

# Checks if the user input is between 1 & 10 and outputs a relevant message

num1 = input(“Enter a number between 1 and 10”)

if num1 >= 1 and num1 <= 10:

print(“Number in range”)

else:

print(“Number not in range”)

# Same as above, but using < & > instead of <= & >=

num1 = input(“Enter a number between 1 and 10”)

if num1 > 0 and num1 < 11:

print(“Number in range”)

else:

print(“Number not in range”)

To check if a number is outside a range, we use:

Less than the lowest value **OR** greater than the highest value.

Only one condition has to be true to ensure that the number is outside the range. It would be impossible for both conditions to be true

# Checks if the user input is not between 1 & 10 and outputs a relevant message

num1 = input(“Enter a number between 1 and 10”)

if num1 < 1 or num1 > 10:

print(“Number not in range”)

else:

print(“Number in range”)

# Same as above, but using <= & >= instead of < & >

num1 = input(“Enter a number between 1 and 10”)

if num1 <= 0 and num1 >= 11:

print(“Number not in range”)

else:

print(“Number in range”)

#### Tasks

##### Predict And Run

Task and instructions - <https://repl.it/@MrAColley/234-Range-Validation-Predict-Run>

Example solution - <https://repl.it/@MrAColley/234-Range-Validation-Predict-Run-Solution>

##### Investigate And Modify

Task and instructions - <https://repl.it/@MrAColley/235-Range-Validation-Investigate-Modify>

Example solution - <https://repl.it/@MrAColley/235-Range-Validation-Investigate-Modify-Solution>

##### Make

Task and instructions - <https://repl.it/@MrAColley/236-Range-Validation-Make>

Example solution - <https://repl.it/@MrAColley/236-Range-Validation-Make-Solution>

### 

### 

### 

### 

### 

### 

### 

### 3 - Nesting

#### TEACHER NOTES

Nesting is the term for putting programming constructs of the same type inside each other.

These examples use nested selection - if statements inside other if statements.

Replit makes it easier to track nesting by putting vertical lines in the code that run between the parts of the same statements.

Students can easily get confused between the statements, get them to trace vertically to help them understand the flow.

The nested statement will only be run if the condition above it is true.

For example:

if num1 > 20:

print("Bigger than 20")

# nested if is indented so is part of the ‘if’ statement above.

if num1 > 50:

print(" and bigger than 50")

else:

print("but not bigger than 50")

#### Tasks

##### Predict And Run

Task and instructions - <https://repl.it/@MrAColley/237-Nesting-Predict-Run>

Example solution - <https://repl.it/@MrAColley/237-Nesting-Predict-Run-Solution>

##### Investigate And Modify

Task and instructions - <https://repl.it/@MrAColley/238-Nesting-Investigate-Modify>

Example solution - <https://repl.it/@MrAColley/238-Nesting-Investigate-Modify-Solution>

##### Make

Task and instructions - <https://repl.it/@MrAColley/239-Nesting-Make>

Example solution - <https://repl.it/@MrAColley/239-Nesting-Make-1>