Week 3 - Selection

### General Teacher Notes

#### What is Selection?

This week is focused on **selection,** the code that makes the computer perform different tasks in different situations. Selection is sometimes referred to as a decision, and this often helps when we are explaining the idea to students. Whenever we make an IF *this* THEN *that* type decision we are using selection logic. There are a lot of concepts to unpack in even a simple selection statement. I’ve tried to break it down and build up slowly and often find that it takes my students lots of time & practice to truly master them.

For example, real life decisions could be:

*IF it is raining THEN*

*Take your umbrella.*

Or:

*IF you haven’t done your homework THEN*

*You get grounded.*

In both of these examples, the instruction on the second line is only followed if the scenario on the first line is *true.* Otherwise it is skipped over. This is an essential part of selection explained in more depth below.

#### Boolean Conditions

Whereas in all of the code we have written before the computer runs each line one at a time, selection relies on a **Boolean Condition** to decide whether or not to run the code.

Boolean conditions are true or false checks. If a condition is checked and comes back as **true**, the code on the line(s) below runs. If it comes back as **false**, then the computer skips the commands and moves on with the rest of the program.

Let’s put that into practice with the umbrella example above:

IF **it is raining** THEN

Take your umbrella

The condition is in red. It is either raining (True) or not (False). If it is raining (true) then the instruction to take your umbrella is carried out. If not (false) then we would skip down to the next instruction that wasn’t part of the selection (probably something like ‘leave the house’).

#### 

#### Boolean Operators

Boolean conditions are written using logical operators. These are:

== Equal to/Same as (a single = is used to store data in a variable, so Python uses double = to compare two pieces of data)

!= Not equal to

The above two conditions can be used with integers or strings. The ones below can only be used with numeric data (integers or floats/decimals)

> Greater than

>= Greater than or equal to

< Less than

<= Less than or equal to

Let’s use the umbrella example again. We want to compare the weather to rain to see if they are the same, so we would write our condition like this:

IF weather == “rain” THEN

Take your umbrella

**Case matters** when comparing text using ==. The match has to be **exact**. For the example above, if the user inputs ‘rain’ then the condition will be true. If they input ‘Rain’ then it will be false. Future units will look at how to get around this issue.

The examples above only refer to selection for one situation. See the lesson slides and individual task notes for how to develop this for two or more situations.

#### 

### 1 - Selection With One Outcome

TEACHER NOTES These selection examples either output a statement (if the condition is true) or they appear to do nothing (if the condition is false). In reality the computer is just skipping the indented line(s) below the ‘if’. Because there is no more code after this, the computer has no more instructions to execute so completes the program. Students should only get an output if their condition is true.

To summarise, this code should be written like this:

*if condition:*

*Code to run if the condition above is true.*

Task and instructions - <https://repl.it/@MrAColley/31-Selection-With-One-Condition>

Example solution - <https://repl.it/@MrAColley/31-Selection-With-One-Condition-Example-Solution>

# Task 1 - Add comments to explain what the input line does, and when the program below will output the text 'This text is output because the condition was true'

num1 = int(input("Enter a number"))

if num1 == 10:

print("This text is output because the condition was true")

# Task 2

# Edit the program below so that it works properly for the correct legal driving age. Add comments to explain what the Boolean operator means. Change the value in the age variable to

age = 18

if age > 18:

print("You are old enough to drive")

# Task 3

# Write a program that asks the user to input their name. If they enter 'Dave' then the program outputs 'Hello Dave'

### 2 - Selection With Two Outcomes

TEACHER NOTES These selection examples output one of two different statements. The indented code below the ‘if’ runs when the condition is true. The ‘else’ is used as a catch all - if the condition is false then the computer skips down to the else and runs the indented code there instead. **Please note -** the else does not have a condition. It doesn’t need one because it runs every time the condition after the ‘if’ is false.

To summarise, this code should be written like this:

*if condition:*

*Code to run if condition above is true.*

*else:*

*Code to run if condition is false.*

Task and instructions - <https://repl.it/@MrAColley/32-Selection-With-Two-Outcomes>

Example solution - <https://repl.it/@MrAColley/32-Selection-With-Two-Outcomes-Example-Solution>

# Task 1

# Add comments to the code to explain:

# What will be output when the code is run?

# In what circumstances would the other output message be produced

num1 = 42

if num1 == 42:

print("You have discovered the meaning of life!")

else:

print("Sorry, you have failed to discover the meaning of life!")

# Task 2

# Add to the code below so that it outputs 'You're not Dave!' if the user does not input 'Dave'

name = input(“What’s your name?”)

if name == "Dave":

print("Hello Dave")

#EXTRA CHALLENGE - Adapt the code so that it works in the same way but uses a not equal to Boolean operator.

**TEACHER NOTES FOR TASK 2**

Students might complain that they are typing ‘Dave’ but not getting the ‘Hello Dave’ output. Double check that they are using capital D (case has to match exactly) and not pressing space before hitting enter. Either of these will mean that the input is not an exact match for ‘Dave’.

# Task 3 - The login checker

# Write a program that:

# Stores the number 1337 in a variable called 'password'

# Asks the user to guess the password and stores their input in a new variable (you choose the name)

# If the user inputs 1337 then output 'Password correct', otherwise output 'Password incorrect'

# Task 4 - Biggest number

# Write a program that:

# Asks the user to input two different numbers and stores them in two variables.

# Outputs the biggest number entered

**TEACHER NOTES FOR TASK 4**

Students often forget to get the inputs one at a time and store them in two separate variables. See the example solution for how to do this.

Get students to run this code several times to test it. Some may spot that it doesn't work properly for two numbers that are the same. For that we need a third outcome, which leads us nicely on to the next set of activities.

### 3 - Selection With Three or More Outcomes

TEACHER NOTES These selection examples output one of three or different statements. The indented code below the ‘if’ runs when the condition is true. The ‘elif’ is short for ‘else if’ - this allows us to add another condition to be checked if the previous one is false. You can add as many of these as you like defending on how many outcomes you need. The ‘else’ comes last and is used as a catch all - if the other conditions are false then the computer skips down to the else and runs the indented code there instead.

To summarise, this code should be written like this:

*if condition:*

*Code to run if condition above is true.*

*elif condition:*

*Code to run if condition above is true.*

*|*

*|*

*As many elifs as necessary (two less than the total number of outcomes - one outcome is covered by the if, another by the else).*

*|*

*|*

*else:*

*Code to run if all conditions have been checked and found false.*

NB - you don’t *need* to include an else at the end of a selection statement, but it is usually good practice, and there’s enough cognitive load in this session already without introducing another element. I’ve found it useful to teach beginner students that they should always include an else.

Task and instructions - <https://repl.it/@MrAColley/33-Selection-With-Three-Or-More-Outcomes>

Example solution - <https://repl.it/@MrAColley/33-Selection-With-Three-Or-More-Outcomes-Example-Solution>

# Task 1

# Add comments to explain the overall job of the program.

# Add comments to the code to explain the circumstances that would produce each output

num1 = (int(input("Enter a number")))

num2 = (int(input("Enter another number")))

if num1 > num2:

print (str(num1) + " is bigger.")

elif num1 < num2:

print (str(num2) + " is bigger")

else:

print("The numbers are the same")

# EXTRA CHALLENGE - Why do we not need a condition after the 'else'?

# Task 2

# Add to the program so that if the user inputs 'Music' they get a message saying 'Not bad, but Computing is better'. For all other subjects the user should get a message saying 'How wrong can you be? Computing is waay better than that!'

subject = input("What's the best school subject?")

if subject == "Computing":

print("That is the right answer!")

elif subject == "computing":

print("You are correct, but Computing starts with a capital 'C' ")

# Task 3 - Which room

# Write a program that asks the user for their name and which subject they are studying.

# The program should output a message telling the student by name which room to go to for that class (make up the room numbers if you need to). You should include at least 3 subjects and have a message such as 'I don't know which room that class is in' for any you don't include.

# Example - for input of 'Ben' and 'Computing' might get an output of 'Hi Ben, go to room 401 for Computing'

### Homework Task

Example solution - <https://repl.it/@MrAColley/34-Insult-O-Matic-Example-Solution>

Write a program that:

* Asks for the user’s name.
* Asks the user to input a number between 1 and 5.
* Outputs a personalised insult (that includes the user’s name) depending on which number they picked.

Keep your insults clean!