

Python 8: Indeterminate Loops

Teaching Resource

Resources

- Slides **Python 8: Indeterminate Loops**
- You will need to either have Python IDLE installed or have access to an online Python IDE. We have used <https://editor.raspberrypi.org/en>
- Activity worksheets are included in this lesson. You will need to distribute these to your pupils
- We have added a walk-through video below

Prior Knowledge

Printing, mathematical operators, data types and input using Python

- Selection in Python using `if..elif..else`
- counted `while` loops in Python
- Selection using `if..elif..else` in a `while` loop



3MB

Python 8 - Indeterminate Loops.pptx

Lesson walk-through

You need to log in to access videos



Learning Objectives

To teach pupils to:

- understand the differences between counted loops and indeterminate loops
- use indeterminate loops in Python using `while`

Slide Notes

Starter

Slide 2: Show the code and ask pupils to work out what it will do, given different inputs. You might like to ask pupils to discuss in pairs, as suits your class. If pupils are unsure, ask them to suggest different possible inputs and work out what they will each output in order to help them notice the pattern.

Slide 3: Show the answers and ensure that pupils understand the reason. This loop will continue to ask "Are we there yet?" until the user enters "yes". The input must

be in lowercase.

Activity 1: Exploring indeterminate loops in Python Den

Slide 5: Show this slide (which is the map for level 25 in Python Den) and ask pupils how they can use the blocks shown to solve the level. Discuss this differences between this solution and the counted loops we have been using up to this point. With this type of loop (an indeterminate loop), we don't need to know how many times the loop will repeat when we write the code.

Slide 6: This slide shows the solution in block and Python code. Discuss both of these and ensure that pupils can relate the blocks to the Python. Once pupils are ready, ask them to do levels 25–30 in Python Den. On level 30, they need to type the Python code directly, so you might like to keep slide visible.

Activity 2: Selection in an Indeterminate loop in Python Den

Slide 8: Show this slide (which is the map for level 31 in Python Den) and discuss how it can be solved. It is necessary to use `else` for this level. The solution follows on slide 9.

Slide 9: Show the solution and discuss it with your class.

Slide 10: Pupils should do levels 31–40. These levels include `if..else`, `if..else if..else`, cows (levels 38 and 39) and traffic lights (level 40)

Activity 3: Exploring indeterminate `while` loops

Slide 12: Pupils should use the provided model code to answer questions 1–2. Those who finish quickly can go on to do the extension. Answers are provided on slides 13–16. You might like to use slides 8–9 to work through a solution to the extension with your class.

Activity 4: Validation

Slide 18: Validation is the process of checking that input is reasonable before allowing it to be stored and used in a program. Its purpose is to avoid runtime errors and bad data in a system. The code example shows that any input other than "yes" is treated as "no". We could use `elif` to capture "no" and then `else` to capture anything else, but that is not enough, as we want to repeat the input if they don't enter "yes" or "no".

Show this slide to pupils and ask for suggestions as to how we could add validation to this code. At this point, you are only looking for ideas, not necessarily how that could be implemented in the code.

Slide 19: Show the code and ask pupils which part of the code needs to be repeated until a valid answer has been input. The answer is, just the input.

Slide 20: This slide gives the answer to the question posed on slide 19 and asks pupils to try this in Activity 4.

There are lots of ways of completing this program, and two suggested answers are shown on slides 21–22.

Slide 21: We could keep repeating the input until "yes" or "no" have been entered by changing the while loop to `while answer != "yes" and answer != "no"` but, if we do this, we must set `answer` to something before the `while` loop or the program will crash as `answer` will not have been defined. Ask pupils what `answer` should be set to. The answer to this is that it can be set to anything apart from "yes" or "no" and the sensible choice is an empty string.

Slide 22: This slide shows an alternative, an arguably better, solution which uses a Boolean to control the `while` loop. In this solution, we control the loop with a Boolean variable, `valid`. We set `valid` to `False` before the loop and only set it to `True` if the input was "yes" or "no". If we set `valid` to `True` then the while loop will not repeat again because the condition `not valid` is not satisfied.

i It is better to use `not valid` rather than `valid == False`. This is because we do not need to compare `valid` to anything as it is already a Boolean value. `not valid` negates what is in `valid` so if `valid` was `True`, `not valid` will be `False`, and vice versa.

We could have put the `if` statement inside the loop, but it is somewhat clearer to have it after the loop. This code is only reached when the loop finishes.

Plenary

Slide 23: The exit ticket shows a Python program and asks pupils to work out what the given program does. There is no need for them to run the code, this is a good exercise in code reading. Ask pupils to discuss this in pairs and, if they are stuck, ask them to try dry running it with specific inputs.

Solution: The loop keeps repeating whilst the `total` is less than 100. Each time the loop repeats, the user is asked to enter a floating point value and this number is added to `total`. Once the loop completes, the total of all weight values entered is output. You might like to ask the class how many times the loop is repeated with specific inputs, e.g.

- 100 - answer: the loop repeats once
- 99.5, 0.2 etc.
- However, if you enter small numbers, the loop will repeat lots of times until their total is at least 100