

Python 7: Selection in a Loop

Teaching Resource

Resources

- Slides **Python 7: Selection in a Loop**
- 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



3MB

Python 7 - Selection in a loop.pptx

Lesson walk-through

You need to in to access videos



Learning Objectives

To teach pupils to:

- practice what they have learned about `while` loops so far in Python exercises using Python Den and Python on its own
- to write more complex programs by using prior solutions as scaffolds
- to use selection inside a loop
- to use Boolean operators in more complex selection statements

Slide Notes

Starter

Slide 2: Show the code and ask pupils to work out what it will display when the specified values are input. You might like to ask pupils to write this down or discuss in pairs, as suits your class.

Slide 3: Show the answers and ask pupils to correct their own answers.

Activity 1: Selection in a `while` loop in Python Den

Slide 5: Show this slide and ask for suggestions as to how to fix this code. Slide 6 shows the available blocks, this might give some a hint.

Slide 7: Show this slide and ask pupils to suggest how to finish the code using these additional blocks. They should then attempt level 14. The solution follows on slide 8.

Slide 8: Show and discuss the solution, and then ask pupils to complete levels 15–18 on Python Den for practice. Levels 16–18 require pupils to type in the Python, so you might want to show the syntax on the screen.

Activity 2: Exploring `if..else` in a `while` loop

Slide 11: Show this slide and the suggested solution to the route shown. Ask pupils if this code will work. It will not, as there is not always a right-left turn after a forward.

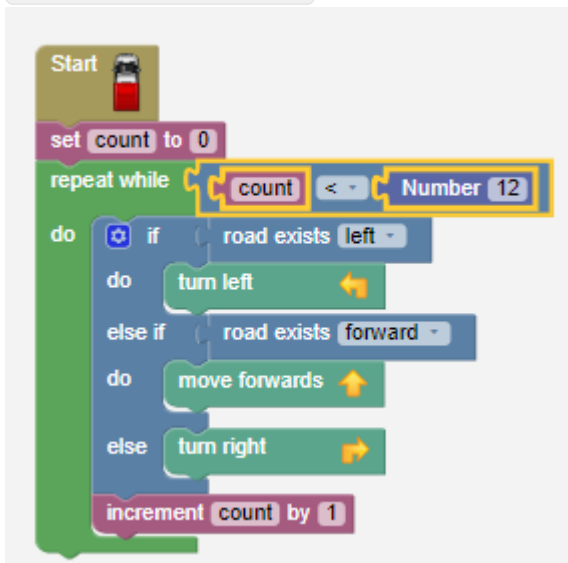
Slide 12: Ask pupils what they need here. You might like to hint that they need to offer an alternative to moving forward.

Slide 13: Show the solution using `else`, the Python code follows on slide 14. Discuss any queries about this code. Ensure that pupils understand that it is more efficient to use an `if..else` rather than two `if`-statements. If you use two `if`-statements, then you ask unnecessary questions. Once pupils are ready, ask them to complete levels 19–21.

Activity 3: Exploring `if..elif..else` in a `while` loop

Slide 16: Show this route and ask pupils to point out which ways the van needs to go to get to its destination. The answer is left, right and forwards at different times.

Slide 17: Show this partial solution. You might need to remind pupils how to get the `if..else if..else` blocks in Python Den. The solution is



Ask pupils to complete level 22 and then continue to level 25.

Activity 4: Exploring selection in a `while` loop

Slide 19: Pupils should use the provided model code to answer questions 1–3. Those who finish quickly can go on to do the extension. Answers are provided on slides 20–25. You might like to use slides 23 - 25 to work through a solution to the extension with your class.

Activity 5: Counting even numbers

Slide 27: Show the code and allow pupils to explore the code as necessary. You might like to ask them to discuss what it does in pairs, run it and explore it, trace it by hand or discuss it line-by-line as a class.

Maths support: You might want to review the modulus operator with your class and ask them what the remainder will be if one number is divisible by another number, e.g. $12 \% 2$ gives 0; $25 \% 5$ gives 0

Once pupils are comfortable with the code and why it produces the given output, ask them how they could extend this code so that it counts how many even numbers were entered. You might like to ask pupils to discuss this in pairs and suggest solutions.

Slide 28: Show this slide and ask pupils to suggest where the provided lines of code should go. You are likely to find that many pupils will not get this right, so allow plenty of time for this discussion. Allow pupils to try this for themselves; it is Activity 5 on their worksheets.

Slide 29: Show this slide, which gives the solution and highlights the reasons for the placement of each line. Allow pupils to correct their code, if necessary.

Activity 6: Boolean Operators

Slide 31: Ask pupils to do the prediction exercises on their worksheets. It is important that they do not run the code at first, but instead take time to read it and try to work out what it will output. Once they have answered the questions on the worksheet, show slides 32–33 and discuss the output for each exercise.

Activity 7: Boolean Operators in a Loop

Slide 35: Ask pupils to do Activity 7. They should use the solutions for the prior activities to help them.

Slide 36: This slide highlights a common error with the use of Boolean operators. It is essential that each expression between `and` or `or` is a Boolean expression, i.e., that it evaluates to `True` or `False`. Forgetting to do this can result in some bugs that are tricky to spot. The code below would always evaluate to `True`, regardless of what is input because any non-empty string in Python is considered `True` in a Boolean context.

```
colour = input("Enter a colour ")
if colour == "red" or "blue":
    # Do something
```

Plenary

Slide 37: The exit ticket shows a Python program and asks pupils to work out why it does not work. 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, give them the hint that the error is on line 5.

Solution: The `if`-statement is incorrect. The colour cannot be red AND blue AND green, the Boolean operator used here must be `or`