

Python 3: Selection

Teaching resource

Resources

- Slides **Python 3: Selection**
- 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 walkthrough video of the lesson below

Prior Knowledge

Printing, mathematical operators, data types and input using Python

Vocabulary

- **code block** - a group of statements that are grouped together
- **condition** - a Boolean expression used to decide what happens next in a program
- **evaluate** - to work out the answer of an expression, e.g. `round(10.125,2)` evaluates to `10.13`
- **indentation** - leaving spaces (usually 4) before you type more code. Indentation in Python is used to indicate what is in a code block



2MB

Python 3 - Selection.pptx

Lesson Walk-through



You need to be logged in to access



Learning Objectives

To teach pupils to

- Use `if..elif..else` to make decisions in Python programs
- Understand the significance of indentation in Python

Slide Notes

Lesson Introduction

Show slide 3 on the interactive white board and ask pupils to look at the flowchart and try to work out the answers to the questions. You might like to ask them to do this in pairs.

Even if pupils are not familiar with flowcharts, ask them to follow the boxes and directional arrows. These symbols are the same as those used at GCSE.

Ask pupils to review the relational operators on their worksheets before moving on to the next slide.

Once you have correct answers to these questions, show the next slide. On this slide we see a Python program that does the same as the flowchart. Ask pupils what they notice about this code that might be unfamiliar. You are looking for comments about the indentation and the colon. Slides 5 - 6 highlight these important points. Take care to use this terminology: indentation, code block, evaluate and condition.

Activity 1: Indentation

Slide 8: Ask pupils to predict what the code will output for the given values. They should not run the code yet, it is important for them to think about their expectations first, even if they are wrong. Once they have written down their predictions, they should type in the code to find out the answers.

Slides 9–11: Ask pupils to explain the output shown before moving on.

Slide 9: The input colour is equal to "red" so the code in the indented block is run, outputting "I like red too!" and "The world's biggest flower is red". Then the unindented code is run, outputting "Have a nice day"

Slide 10: "Blue" is not the same as "red" so the code in the indented block is not run. However, the unindented code is run, outputting "Have a nice day"

Slide 11: Python is case-sensitive, so "RED" is not the same as "red". Therefore, the code in the indented block is not run. However, the unindented code is run, outputting "Have a nice day"

Slides 12–13: Now pupils should use the provided code as a model to complete a different exercise. Scaffolded steps are provided. Solutions are provided on **slides 14–16**.

Slide 14: Note the use of descriptive variable names (e.g. `answer`) and clear input prompts and output messages. Encourage your pupils to use variables that clearly describe what they are being used for.

Slide 15: Note where the `score` is initialised (set to 0 in the first place) and incremented (increased). It must be initialised before the `if` statement, and it must be incremented in the indented block so that it only happens if the answer is correct.

Slide 16: Note that the `score` is output after the `if` statement and is not indented. It is output regardless of whether they got their answer right or wrong.

The use of `if` and `else`

Slide 17: Show this slide and ask pupils which solution is better (they both work).

Slide 18: Whilst both work, solution B is better. If you use the method in A, not only are you asking unnecessary questions, you run the risk of missing some options in more complex examples. More on this later...

The use of `elif`

Slide 19: Show this slide and ensure that pupils understand the table.

Slides 20–22 build up the code slowly.

Emphasise the use of `elif` meaning 'otherwise if'. The code will run the first matching block. So, for example, if the mark was 77, the first `elif` block is triggered and the grade is set to 'Very good'. Then the next line of code to be executed is the last line, `print("The grade is", grade)`, the other `elif` blocks and the `else` block are entirely skipped. This is called a **mutually exclusive condition**.

Activity 2: Complex Conditions use `elif` and `else`

Slide 24: Show the code and ask pupils to try it. They should then do the Activity 2 exercises. The solutions are on **slides 25–26**.

Plenary

Wrap up the lesson with a summary of what has been covered. You might like to ask pupils to explain the use of `if`, `elif` and `else` and the meaning of indentation in Python. You could also ask them for errors that they made, for example, forgetting the colon after the condition.

Show **slide 27** on the interactive whiteboard and ask pupils to write down their answer on mini whiteboards. The correct answer is **Answer 2** and **Goodbye**