

Python 1: Output, Operators and Data

Teaching resource

Resources

- Lesson slides **Python 1: Output, Operators and Data**
- 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

- Experience with a block-based programming tool such as Rapid Router is advantageous but not necessary
- Pupils are not expected to have any textual programming language experience

* Note that different Python IDEs might give slightly different error messages. We have used the error messages from the [Raspberry Pi code editor](#) in these resources.

Vocabulary

- **algorithm** - a finite and ordered set of unambiguous steps to solve a computational problem – for a person
- **case-sensitive** - when the case (upper or lower case) is important
- **modulus** - a mathematical operator that finds the remainder of a division
- **operator (mathematical)** - a symbol that indicates what mathematical function should be applied, e.g. + means add
- **output** - when a computer presents a result to the user, e.g. by displaying on the screen

- **power**- raise a number to the power of another number, e.g. 2 to the power of 3 is 8 ($2 * 2 * 2$)
- **Program** - a sequence of instructions that implements an algorithm – for a computer
- **remainder** - the remainder of a division, e.g. the remainder of 13 divided by 5 is 3



3MB

Python 1 - Output, Operators and Data.pptx

Lesson walkthrough

You need to be logged in to access



Learning Objectives

To teach pupils to

- Use basic programming concepts in Python
- Recognise common errors when using the print statement

- Be able to use Python's mathematical operators
- Understand the purpose and use of different data types

Slide Notes

Introduction

- **Slide 3:** Emphasise to pupils that Python is a widely used language, not only in education but also in many different industries.
- **Slide 4:** It is important that pupils understand that programming is a practical skill requiring logical thinking, persistence and resilience. Learning to play a musical instrument, perfecting a swimming stroke, preparing for a race and honing skills on the sports pitch are good analogies. You might like to ask pupils if they would attempt that stunt without lots of small steps and practice, not to mention lots of failures along the way!
- **Slide 5:** Introduce this terminology if pupils have not yet encountered it, or recap if necessary.
- **Slide 6:** When using Python, pupils must be precise. Spelling mistakes and even incorrect use of case will cause errors.

Activity 1: Hello World!

- **Slide 8:** Demonstrate "Hello World!" and then ask pupils to do this. This is really just to get them used to how to interact with Python.
- **Slide 9:** Some pupils might make some typing errors, so check that everyone has been successful and show them what the result should have been.
- **Slide 10:** Ask pupils what the error is here. Answer is on slide 11: `print` is case-sensitive.

Activity 2: Common Errors

Slide 13: The purpose of this exercise is for pupils to try out the basics of the programming environment. Answers are on slides 14 - 16.

Pupils might have errors if they mistype their code. If they are using an installed IDE such as IDLE, encourage them to read error messages from the bottom up, e.g.

```
>>> Print("Hello world!")
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    Print("Hello world!")
NameError: name 'Print' is not defined
```

Below is an example of the same error message from the Raspberry Pi code editor.

```
NameError: name 'Print' is not defined on line 1
of main.py
```

Error message from Raspberry Pi code editor

Note that Python is case-sensitive so `Print` is not understood, it must be `print`.

Activity 3: Mathematical Operators

Slide 18: Ask pupils to try each of these mathematical operators in turn. The purpose of some should be obvious but others will need some investigation. Encourage pupils to investigate the ones they are unsure of and suggest numbers to try to help them work out the purpose of the mathematical operators on their own. E.g. $14 \% 5$; $15 \% 5$ etc.

- i The `//` operator is **integer division**. This means that you divide the first number by the second but disregard the decimal part or remainder. It does not round, it truncates. E.g. $16 // 5$ is 3 and $19 // 5$ is also 3

The **modulus operator** (`%`) gives the remainder of a division, e.g. $16 \% 5$ is 1 because 16 divided by 5 is 3 remainder 1. This operator is very useful for checking if one number is divisible by another. E.g. $221 \% 13$ is 0 which tells us that 221 is divisible by 13.

Slide 19: Answers

Operator	Example	Output	What does this operator do?
+	<pre>print(17 + 3)</pre>	20	Adds numbers
-	<pre>print(20 - 11)</pre>	9	Subtracts the second number from the first number
*	<pre>print(15 * 3)</pre>	45	Multiplies the two numbers together
/	<pre>print(11 / 5)</pre>	2.2	Divides the first number by the second number
//	<pre>print(11 // 5)</pre>	2	Integer divides numbers (the decimal part is cut off - not rounded)
	<pre>print(17 // 5)</pre>	3	
%	<pre>print(11 % 5)</pre>	1	Finds the remainder of a division. This is called the modulus.
	<pre>print(17 % 5)</pre>	2	
**	<pre>print(10 ** 2)</pre>	100	Raises the first number to the power of the second number
	<pre>print(2 ** 3)</pre>	8	

Activity 4: Data Types

Slide 20: Introduce this key concept of data type. It is important to include the fact that a data type doesn't just impact on what you can store, but also what you can do with what you store. For example, if you have a number, you can do maths with

it. However, if you have a string, you cannot do maths but you can, for example, find out the length of that string.

Slide 22: Having discussed data type, see if pupils can categorise these examples. You might like to show the previous slide to remind them of the rules.

Slide 23: Answers are shown here. Common misconception: not realising that putting quotes around a value makes it a string, no matter what the characters are. E.g., "123" is a string and not an integer.

Value	Data Type
-20	Integer
0	Integer
1.0	Floating point / Real
"100"	String
True	Boolean
"True"	String

Activity 5: Investigation Tasks

Slide 25: The purpose of these tasks is to encourage pupils to investigate the implications of the use of data type and discover some of their impacts in Python.

- Ask pupils to try each of these Python statements and note down what the output is and why it has occurred. Answers are on the following slide.
- The second one `print("10" + "15")` might surprise some pupils. Python has used the + symbol to indicate that strings should be joined (concatenated) and no attempt has been made to mathematically add the values together.
- The third one `print("10 + 15")` is there to explore a common error. Pupils often put values in quotes (making them strings), when they intend something different. This might come up again later when you explore variables or input.

Python will make no attempt to evaluate (work out an answer to) `10 + 15` because this is inside quotes.

Slide 26: Answers follow on slide 26.

Code	Output	Explanation
<pre>print(10 + 15)</pre>	25	This adds two integers
<pre>print("10" + "15")</pre>	1015	Here the + operator joins two strings
<pre>print("10 + 15")</pre>	10 + 15	The numbers are in quotes so they are treated like a string

Plenary

Slide 27: Wrap up the lesson by reminding pupils what they have covered.

The exit ticket is to recall which mathematical operator can be used to find out if one number is divisible by another. You might like to ask pupils to use mini whiteboards to answer this. The correct answer is `%` - the modulus operator. This is useful as it gives the remainder of a division, e.g. `15 % 2` gives 1 and `16 % 2` gives 0. If the result is 0 then it means that the first number is divisible by the second.