

Python 2: Variables, Input and Casting

Teaching resource

Resources

- Lesson slides **Python 2: variables, Input and Casting**
- 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

Prior Knowledge

- Printing in Python
- The use of mathematical operators in Python
- The concepts and usage of data types in Python

Vocabulary

- **cast** - to change a value from one data type to another, e.g. string to integer
- **input** - input is when you ask the user to enter a value which you can then use in your program
- **variable** - a name given to a storage location in memory where you hold data values



3MB

Python 2 - Variables, Input and Casting.pptx

Lesson walk-through

You need to be logged in to access



Learning Objectives

To teach pupils to

- Create and use variables
- Get input from the user
- Convert values between different data types using casting

Slide Notes

Lesson Introduction

Slide 3: Recap the end of the last lesson with the starter. Ensure that pupils are using the correct terminology: data type, operator, operand and concatenation.

Slides 4–6: Reinforce the use of the term 'concatenation'. Note that spaces must be explicitly included, and you cannot use the + operator when the data types of the

operands are different. For example, the third line of code would result in an error because it is attempting to add a string and an integer.

```
code = "ABC"  
number = 123  
result = code + number // You cannot add a string to an integer
```

Activity 1: Investigate the print statement

Slide 8: Ask pupils to complete Activity 1 on their worksheets. They should try each example, note down the output, and answer the question.

The output is shown on **slide 9**. The comma allows a print statement to output multiple things in one line of code. These can be literal values, e.g. "Fred", or variables, e.g. name. Notice that the use of a comma inserts a space at that point.

 Note: It is possible to use print formatting using Python's f-strings to eliminate the unwanted space in the last example. We will come to this in a later lesson.

Key Concept: Variables

Slides 10–11: Pupils sometimes struggle with this concept. The formal definition is that a variable is a label that points to an area of the computer's RAM where a value can be stored. We refer to variables using labels, e.g. name, age, score, rather than memory addresses as they are more user-friendly.

The idea of a box might be helpful to some. Imagine a room full of boxes. You would need them to be labelled if you stood any chance of finding something once you had stored it. However, it is important to note that a variable can only contain one item (until we get on to lists in a later session), so if you put something in the variable, it replaces what was there before.

Activity 2: Variable Naming

Slide 13: Show pupils these rules and conventions. The hard rules are underlined, the others are conventions. Note that your GCSE exam board might have its own rules which must supersede these shown.

Once you have discussed these rules and conventions, ask pupils to do the activity. The answers are also on **slide 14**.

Variable name example	Okay?	Explanation
high_score	✓	
x	✗	Single-letter - this is too short to be descriptive
Surname	✗	Starts with a capital letter
third.place	✗	Has a full-stop in it
the_highest_value_scored	✗	Too long

Activity 3: Printing variables

This activity uses Sentance et al's **PRIMM** framework. You might like to read about it [here](#).

- **P - Predict:** pupils are asked to try to work out what a new piece of code will do, without running it
- **R - Run:** then they run it to see if they were correct and note anything significant
- **I - Investigate:** then they investigate it
- **M - Modify:** then they change it to make it do slightly different things
- **M - Make:** finally, they use models to create something new

The purpose of this task is to model the PRIMM framework and allow pupils to gain confidence with what we have covered so far.

Step 1: Predict

Slide 16: Pupils should work in pairs and discuss what they think each example will do. It is important to emphasise that it doesn't matter if they get their answers wrong; the importance of this is for them to engage with the code and try to work out what it does. They must not cheat by running the code yet!

```
1 name = input("What is your name? ")
  print("Hello", name)

2 hobby = input("What is your favourite hobby? ")
  print(name, "likes", hobby)

3 country = "Kenya"
  country = input("Enter a country name ")
  print("I'd like to visit", country)
```

Slide 16

Step 2: Run

Slides 17–18: Now pupils should run each example in turn and note down what they output. The output is shown on **slide 18**.

Slide 19: use this slide to discuss what the pupils have observed about the `input` command.

Step 3: Investigate

Slide 20: Pupils should try to make some directed changes and note the effect.

These three examples explore some common errors.

```
1 print("Hello", name)
   name = input("What is your name? ")

2 input("What is your favourite hobby? ")
   print(name, "likes", hobby)

3 COUNTRY = input("Enter a country name ")
   print("I'd like to visit", country)
```

The answers are on **slides 21–26**

1. The name is used in a `print` statement before a value has been stored in it
2. The user is asked for input, but it is not stored in a variable.
3. The input value is stored in variable, `COUNTRY`, but we try to print variable, `country`. Python is case-sensitive, so these two variables are not the same as each other.

Step 4: Modify

```
name = input("What is your name? ")
print("Hello", name)

hobby = input("What is your favourite hobby? ")
print(name, "likes", hobby)
```

Slide 27: Ask pupils to change this code to ask for a game name and high score instead. An example solution is also on slide 28.

```
1 game = input("What is your favourite game? ")
2 print(game, "is great!")
3 high_score = input("What is your high score?")
4 print("Great! Your high score on", game, "is", high_score)
```

What is your favourite game?
Plong
Plong is great!
What is your high score?
100
Great! Your high score on Plong is 100

Step 5: Make

Slide 29: Now ask pupils to use this as a model to write a different program which asks about animals. They can decide what to output for this one, though we are a

little limited until we have covered selection (if)!

There is an example solution on **slide 30**.

Activity 4: Casting between data types

Slides 32–33: Start with the investigation task in the pupil worksheet. This allows pupils to discover the issue for themselves: the `input` statement always gives you a string data type and this is not very useful if you want to be able to manipulate numbers. Pupils should try to explain why this has happened before the class discussion prompted on **slide 34**.

Slides 35–36: Discuss and demonstrate the use of `int` and `float` to convert data. Note that converting a number with a decimal part to an integer will result in the program crashing. E.g.

```
1 number = input("Enter a number ")
2 result = int(number)
3 print("The next number is", result + 1)
```

```
ValueError: invalid literal for int() with base 10: '1.2' on line 2 of main.py
```

```
Enter a number
1.2
```

Activity 5: Casting Exercises

Slides 36: Now ask pupils to do the casting exercises. The answers are on **slides 38–40**.

Common errors to look out for

Pupils often get `int()` and `input()` or `float()` and `input()` the wrong way around, e.g.

```
count = input(int("How many millepedes do you have? ")) // Incorrect
```

The reason the code above is incorrect is that it is trying to convert the string "How many millepedes do you have?" to an integer and then use that as an input prompt as the inner brackets are evaluated first.

The correction is:

```
count = int(input("How many millepedes do you have? ")) // Correct
```

This is explored in the exit ticket.

Plenary

Wrap up the lesson by asking pupils for a single sentence to summarise each lesson objective.

The **exit ticket** is to correct the code shown, as it is a common error.

The answer is that `input` and `int` are the wrong way around! It will try to convert the string "Enter a number" to an integer and it will crash. It should be:

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

- ⓘ Remember that you always evaluate the inner brackets first: do the input and then convert what was entered to an integer. The erroneous code tried to convert the string "Enter a number" itself to an integer, which is impossible.