6 - subroutines

### General Teacher Notes

#### What Is A subroutine?

A subroutine is a set of actions that is given a single name (identifier) in a very similar way to a variable or list.

They help us break our program down into smaller individual sections that are easier to test and reuse. This becomes more important as we start to write longer and more complex programs.

A subroutine has to be **defined (**created) before it can be **called** (used) by the program.

There are lots of built in subroutines in Python, but these activities teach us how to create our own. The code for **defining** (creating)a subroutine looks like this:

def subroutineName():

*Code to run when subroutine is called goes here*

return(data) - *this is optional, it is not always needed*

A subroutine can be **called** at any time during the program by typing its identifier. The code for calling a subroutine looks like this:

subroutineName()

This is really useful if your program needs to perform the same task multiple times at different points during the runtime - instead of writing the code for the task multiple times, we would create one subroutine and call it as many times as we need. In this set of lessons we will be creating subroutines that carry out tasks using the skills we have previously learned (input, output, selection, iteration, lists etc).

**Functions**

A subroutine can also return a result (data) to the main program as its final action. This type of subroutine is called a **function.** The code to return data goes **inside** the definition part of the program like this:

return(data)

When the subroutine is called, we usually store the returned data in a new variable. The code for this is below:

newVariable = functionName()

print(newVariable)

**PLEASE NOTE -** a lot of the subroutines in the following tasks are very simplistic. I’ve really broken it down to show *how* to define & call subroutines to reduce the cognitive load on students when learning these techniques. There aren’t really any advantages of coding these tasks as subroutines compared to just coding them into the main program. subroutines become progressively more useful as your programs become more complex.

#### subroutines - Key Concepts/Vocab/Misconceptions

A subroutine is just a ‘package’ for code. You can write any code inside a subroutine, including calling other subroutines (we’ll get on to that bit later).

A subroutine has to be **defined (**created) before it can be **called** (used) by the program.

A subroutine can be **called** at any time during the program by typing its identifier.

subroutines allow us to code common tasks once and reuse them many times. This helps make our program smaller and more efficient.

subroutine names are created using lowercase letters with underscores between words. This is not a syntax rule but it is the common convention. Using this instead of camel case helps us differentiate between subroutines and variables/lists when we are reading the code.

You may hear the terms ‘procedure’ or ‘function’ referred to in other programming languages. In more complex languages these are other ‘flavours’ of subroutine that work in slightly different ways. However, at beginner level Python handily combines them all into subroutines.

A subroutine will not run when it is created (using the **def** command). It has to be called in the program.

A subroutine can have many arguments. They are separated by commas in the brackets when the subroutine is defined.

### 1 - subroutines With Input & Output

#### TEACHER NOTES

These tasks ask the students to predict the output from a program that uses four different subroutines. The subroutines are prewritten and just output one sentence each.

The output does not make sense so they have to rewrite the subroutine call part of the program to put the output sentences in a sensible order.

#### Tasks

Task and instructions - <https://repl.it/@MrAColley/61-Functions-With-Output>

Example solution - <https://repl.it/@MrAColley/61-Functions-With-Output-Example-Solution>

# Task 1

# The program below uses functions. Add comments to predict the output in the order that it has been coded here.

def say\_hi():

print("Why hello there!")

def offer\_drink():

print("Would you care for a spot of tea?")

def offer\_food():

print("Biscuit?")

def say\_bye():

print("Cheerio then.")

offer\_drink()

say\_hi()

offer\_food()

# Task 2

# Rewrite the code to call the functions in the correct order so that the output makes sense chronologically.

# Call the correct function so that the program says goodbye to the user.

# Task 3

# Define a new function that tells the user a joke (you decide on the function name and the joke). Call it in a sensible place in the program.

### 2 - subroutines That Return A Result

#### TEACHER NOTES

subroutines can also return a result (some data) to the main program.

The code for this is:

return(data)

This line of code is usually the last one in a subroutine.

To call a subroutine that returns data, create a variable in the main program and assign the function to it. Here is an annotated example of the whole process.

#Creates & names the function

Def adder():

# Stores two numbers in two variables.

num1 = 10

num2 = 15

# Adds the variable contents together and returns the total to the main program

return num1 + num2

# Calls the adder function and stores the data returned

outputNum = adder()

# Outputs the data in the outputNum variable

print(outputNum)

#### Tasks

Task and instructions - <https://repl.it/@MrAColley/62-Functions-That-Return-A-Value>

Example solution - <https://repl.it/@MrAColley/62-Functions-That-Return-A-Value-Example-Solution>

# Task 1

# Add comments to explain what the output from this program will be and how you know.

def maths1():

num1 = 50

num2 = 5

return num1 + num2

def maths2():

num1 = 50

num2 = 5

return num1 - num2

def maths3():

num1 = 50

num2 = 5

return num1 \* num2

outputNum = maths2()

print(outputNum)

# Task 2

# Adapt the code from one of the functions above to create a new function called 'multiplier'.

# The user should be able to input two numbers that are stored in variables.

# The function should multiply the two variables together and return the result to a variable in the main program.

# The main program should output the variable containing the result returned from the function.

### 3 - subroutines With Arguments

#### TEACHER NOTES

As well as getting data out of subroutines, we can put data in. We do this using arguments. You can think of arguments as variables used by the subroutine. They are named in the brackets after the subroutine name when the subroutine is defined and separated by commas..

In this first example we will put data into the arguments by typing it specifically into the brackets when we call the subroutine. This subroutine has one argument called num1. We will put the number 42 into the subroutine.

def add\_five(num1):

print(num1 + 5)

add\_five(42)

This second example performs the same task, but this time the user can input a number rather than have it fixed to 42. I’ve used a variable called userInput to store what the user types in. The value input by the user gets stored in the *userInput* variable and then **passed** to the *num1* argument when the subroutine is called.

def add\_five(num1):

print(num1 + 5)

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

add\_five(userInput)

We can make this code more efficient by combining the last two lines as shown below. There is a slide for this, but making your students do it the long way first helps them to appreciate the flow of the program.

add\_five(int(input("Enter a number")))

Tasks

Task and instructions - <https://repl.it/@MrAColley/63-Functions-With-Arguments>

Example solution - <https://repl.it/@MrAColley/63-Functions-With-Arguments-Example-Solution>

### Homework Task - Calculator

Example solution - <https://repl.it/@MrAColley/64-Homework-Task-Example-Solution>

Define four subroutines - add, subtract, multiply, divide that add multiply etc two numbers and return the result. Each should have two integer number arguments.

The user is asked to input two numbers. These numbers will be passed as arguments into one of the subroutines.

The user is asked to input 1 to add, 2 to subtract etc.

If they input 1, call the ‘add’ subroutine, input 2 calls the ‘subtract’ subroutine etc

Output the returned result as part of a sentence.