Computing

Lesson 1: Subroutines

Programming Part 4: Subroutines

Ben Garside

 Materials from the Teach Computing Curriculum created by the National Centre for Computing Education

Task 1 - Subroutines

Subroutines worked example

```
def calculate(a, b):
    answer = a + b
    print(f"{a} + {b} = {answer}")

print("Enter a number:")
num1 = int(input())
print("Enter another number:")
num2 = int(input())

calculate(num1, num2)
```

Task 1.1 - The average number - 1

Open a new Repl.it file or create a new Python file on your device

Step 1:

Define a subroutine called average_value. The `average_value` subroutine should accept 3 parameters, `a, b` and `c`.

Step 2:

The purpose of the `average_value` subroutine is to calculate the average of 3 numbers and output them for the user.

Create a variable called `average` and assign to it an expression that will calculate the average of the 3 values passed through the parameters.

Tip: To calculate the average you must first add the three numbers together and then divide that by 3.

Task 1.1 - The average number - 2

Step 3:

After the average has been calculated, the program should display the output:

```
The average value is {average}
```

Step 4:

Test your subroutine by calling it. You can call your subroutine by adding the line average_value(6,8,10) to the end of your program.

If your program is working correctly then it should display the following when it is executed:

```
The average value is 8.0
```

Task 1.1 - The average number - 3

Step 5:

Delete the testing line of code.

Create 3 user prompts that will ask for the 3 numbers. The answers should be held in 3 variables: `num1`, `num2` and `num3`.

Step 6:

Create a subroutine call that will pass the arguments `num1, num2, num3` to the `average_value` subroutine.

Task 1.1 - The average number - 4

Step 7:

Test your program. Use the table below to make sure that your program is working correctly.

Example: (✔ if it was successful)	
Note: Use this example to check your program. This is the output your program should produce for the given input.	✔
The user is given a prompt	`Enter an number:`
The user enters their reply and it is held in a variable.	`4`
The user is given another prompt	`Enter another number:`
The user enters their reply and it is held in a variable.	`8`
The user is given another prompt	`Enter another number:`
The user enters their reply and it is held in a variable.	`10`
The three variables are passed as arguments to the subroutine. The subroutine calculates the average value and the value is displayed.	`The average value is 7.333333333333333`

Task 1.2 - The highest number - 1

Step 1:

Define a subroutine called highest. The `highest` subroutine should accept 2 parameters, `a` and `b`.

Tip: use the worked example on slide 2 to help you structure your subroutine.

Step 2:

The purpose of the `highest` subroutine is to output the highest number passed into it.

Create an `if-else statement` with a condition to check if one value is **higher than** the other.

The highest value should be held in a variable called `highest_num.`

Tip: Look at previous programs that you have created where one value has been compared with another.

Task 1.2 - The highest number - 2

Step 3:

Create a print statement that will output `The highest number entered is {highest_num}` directly after the `if-else` statement.

Step 4:

Test your subroutine. In order to test your subroutine you will need to call it. You can call your subroutine by adding the line `highest(8,2)` to the end of your program.

If your program is working correctly then it should display the following when it is executed:

```
The highest number entered is 8
```

Tip: Check your indents. The subroutine call should be all the way to the left.

Tip: If your program was unsuccessful then you should refer to the worked example and other programs that you have used to compare one value with another.

Task 1.2 - The highest number - 3

Step 5:

Delete the testing line of `code highest(8,2)`

Create a user prompt that will ask for a number. The response should then be held in a variable called `num1`.

Create another user prompt that will ask for another number. The response should then be held in a variable called `num2`.

Tip: see the worked example on slide 2 for help with this step.

Task 1.2 - The highest number - 4

Step 6:

Create a subroutine call that will pass the arguments `num1` and `num2` to the highest subroutine.

Tip: use the subroutine call from step 4 as a guide for this.

Task 1.2 - The highest number - 5

Step 7:

Test your program. Use the table below to make sure that your program is working correctly.

Example: (✔ if it was successful)

Note: Use this example to check your program. This is the output your program should produce for the given input.		✔
The user is given a prompt	`Enter an number:`	
The user enters their reply and it is held in a variable.	`4`	
The user is given another prompt	`Enter another number:`	
The user enters their reply and it is held in a variable.	`8`	
The two variables are passed as arguments to the subroutine. The subroutine checks for the highest value. The highest value is displayed.	`The highest number entered is 8`	

Task 2 - Calculation checker

Worked example

```
def calculate(a, b):
    answer = a + b
    print(f"{a} + {b} = {answer}")

print("Enter a number:")
num1 = int(input())
print("Enter another number:")
num2 = int(input())

calculate(num1, num2)
```


Task 2 - Calculation checker

Scenario:

A teacher would like you to create a program that will help learners check their answers to a maths quiz that they have been given. Each question involves 2 numbers. The 2 numbers could be used for addition, subtraction, multiplication or division. Here are some questions that the learners were given:

1 + 3 = ?

8 x 2 = ?

3 - 1 = ?

8 ÷ 2 = ?

The program will need to:

- Ask for 2 numbers
- Ask if they wish to add, subtract, multiply or divide
- Provide the correct answer e.g. 1 + 3 = 4

Note: At this stage, the program only needs to work for 1 calculation.

Task 2 - Decompose the problem

Step 1:

Think about the subroutines that could be created for this program. Make a list of them below:

Task 2 - Decompose the problem

Step 2:

Think about the other parts of the program. What programming techniques will be required to input the numbers? How will the program make decisions about which subroutine to call?

Write down your thinking in the space provided below:

Task 2 - Calculation checker - 1

Step 3:

Design the algorithm for this program below using pseudocode (oaknat.uk/comp-ks4-pseudocode).

Task 2 - Calculation checker - 2

Step 4:

Create the program based on the pseudocode that you have designed. Here is some sample output to help you test your program.

Example: (✔ if it was successful)

Note: Use this example to check your program. This is the output your program should produce for the given input.		✔
The user is given a prompt	`Enter a number:`	
The user types in a number	`10`	
The user is given a prompt	`Enter another number:`	
The user types in a number	`6`	
The user is given a prompt	`Would you like to +, -, / or * ?`	
The user enters a response	`*`	
The program checks the condition and calls the multiply subroutine. It multiplies the two values given and displays a message.	`10 * 6 = 60`