# Scripts and Modules

## Exercises

### Week 5

Prior to attempting these exercises ensure you have read the lecture notes and/or viewed the video, and followed the practical. You may wish to use the Python interpreter in interactive mode to help work out the solutions to some of the questions.

Download and store this document within your own filespace, so the contents can be edited. You will be able to refer to it during the test in Week 6.

Enter your answers directly into the highlighted boxes.

For more information about the module delivery, assessment and feedback please refer to the module within the MyBeckett portal.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## ©2021 Mark Dixon / Tony Jenkins

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

When a Python program is stored within a text file (i.e. a *script*), what suffix should be used for the filename?

*Answer:*

The suffix used for a python program in text files is with a .py at the end.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Is it necessary to use a special Integrated Development Environment (IDE) to write Python code in text files?

*Answer:*

No, it is not necessary as python can be written in any text editor.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

When a *script* is executed from a file, are the results of evaluating expressions automatically displayed on the screen without the need of a print() function call?

*Answer:*

No, if the script is executed from a file then the print() function needs to be called. However, in interactive mode, it uses REPL so the print function is not necessary.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

What command would need to be typed in an operating system terminal window in order to execute a Python script called PrintNames.py?

*Answer:*

To execute the python script the command would be: python PrintNames.py.

What command would need to be typed in a terminal in order to pass the values "John", "Eric", "Graham" as *command line arguments* to the PrintNames.py script?

*Answer:*

The command line arguments would be: python PrintNames.py John Eric Graham.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

When a Python script wishes to access *command line arguments*, what **module** needs to be imported?

*Answer:*

The module that can access command line arguments is the sys module.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

What is the data-type of the sys.argv variable?

*Answer:*

The data type of sys.argv is a list. It is a list of string values.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

What is stored within the first element of the sys.argv variable?

*Answer:*

The first element of the sys.argv is the program name.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Use a text editor to write the *script* called PrintNames.py. This should display any *command line arguments* that were passed during execution.

Once complete, place your solution in the answer box below.

*Answer:*

import sys

arg = sys.argv[1:]

for value in arg:

print(value,end = " ")

Improve the solution so it uses an if statement to check that at least one name was passed, or otherwise print a message saying “no names provided”. Place your improved solution in the answer box below.

*Answer:*

import sys

arg = sys.argv[1:]

if len(arg) >= 1:

for value in arg:

print(value,end = " ")

else:

print("no names provided")

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

When using an import statement it is possible to provide an *alias* that can be used as an alternative name to access module content.

Write an **import** statement that imports the whole of the sys module, and renames it to my\_system.

*Answer:*

import sys as my\_system

This renames the sys module to an alias called my\_system that can be used as sys.

Write a **from..import** statement that imports only the math.floor function, and renames it to lower

*Answer:*

from math import floor as lower

This imports only the floor function from the math module and renames it to lower.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

What is stored in a *symbol-table*?

*Answer:*

Variable, functions and names used in the module or program and other identifiers are stored in a symbol-table. They do not in

terfere with other modules or the main module.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Why is the following type of import statement generally not recommended?

from math import \*

*Answer:*

This is not recommended as there is a possibility that similar variable already exists in the program that may lead to confusion or clash between them.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

When working in *interactive-mode* what convenient function can be used to list all names defined within a module?

*Answer:*

The dir() function can be used to list all names defined within a module.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

What is the value stored within the sys.path variable used for?

*Answer:*

The sys.path searches its list of directories for the file to be loaded if the name of the module is not known.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

When a program is being executed as a *script* what value is assigned to the special variable \_\_name\_\_?

*Answer:*

The value “\_\_main\_\_” is assigned to the \_\_name\_\_ variable if it is running as a script.

\_\_name\_\_ == “\_\_main\_\_”

What value is assigned to the \_\_name\_\_ variable when a program has been imported as a *module*?

*Answer:*

The value \_\_name\_\_ is assigned the name of the .py file when it is imported as a module.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Why is it useful for a program to be able to detect whether it is running as a *script*, or whether it has been imported as a *module*?

*Answer:*

It is useful as the program can be used both as an executable script or an imported module depending on the situation.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## **Exercises are complete**

Save this logbook with your answers. Then ask your tutor to check your responses to each question.