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.

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

_____

_____

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

Answer:

.py

_____

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

Answer:

No it is not necessary.

_____

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 print() is required.

_____

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

Answer:

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:

Python PrintNames.py "John" "Eric" "Graham"

_____

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

Answer:

Sys module

_____

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

Answer:

 List data type.

_____

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

Answer:

 The name of script.

_____

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
 for arg in sys.argv[1:]:
    print(arg)
```

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:

```
mport sys
 if len(sys.argv) > 1:
    for arg in sys.argv[1:]:
       print(arg)
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

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

Answer:

From math import floor as lower

_____

What is stored in a *symbol-table*?

Answer:

Name, type, memory location of identifiers of a program

_____

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

```
from math import *
```

Answer:

This can cause conflicts and confusion while using various modules

_____

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

Answer:

Dir()

_____

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

Answer:

Path of directories

_____

When a program is being executed as a *script* what value is assigned to the special variable `__name__`?

Answer:

'__main__'

What value is assigned to the `__name__` variable when a program has been imported as a *module*?

Answer:

'__name__'

_____

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 enables the script to include code that executes specifically when run independently, not when imported as a module, promoting modularity and code reusability.

_____

Exercises are complete

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