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.

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

Answer:

.py suffix

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 it wont be displayed automatically. print() function is a must for displaying content on the screen

---

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:

python3 PrintNames.py John Eric Graham`

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

Answer:

sys model needs to be imported to access command line arguments

---

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

Answer:

list

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

Answer:

The first element is the name itself 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:

```
# PrintNames.py

import sys

# The first element of sys.argv is the script name itself
# The following elements are the command line arguments
arguments = sys.argv[1:]

# Display the passed arguments
for arg in arguments:
    print(arg) if arguments else print("No command line arguments provided.")
```

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:

```
 PrintNames.py
import sys

# The first element of sys.argv is the script name itself
# The following elements are the command line arguments
arguments = sys.argv[1:]

# Display the passed arguments
if arguments:
    print("Command Line Arguments:")
    for arg in arguments:
        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:

Information about indentifier found in the source code

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

```
from math import *
```

Answer:

As this imports all the function and variables from math module and it can lead to code conflict
. So it is generally not recommended

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

Answer:

dir() functioin

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

Answer:

A list of directories that the operating system searches when executing a command.

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

Answer:

string value __main__

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

Answer:

The special variable __name __ is assigned the name of the module, not __main_

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:

Code reusability, Debugging, Organization

Exercises are complete

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