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:

We should use “.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 to use a special Integrated Development Environment (IDE) to write Python code in text files. Any generic text editor can be used.

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, Python does not automatically print the results of expressions to the screen without explicit instructions.

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. Syntax: import sys
arguments = sys.argv

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

Answer:

The sys.argv variable data type is a list of strings

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

Answer:

The first element of the sys.argv variable contains the name of the Python 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

print("Names:", sys.argv[1:])
```

Then in cmd we type: **python PrintNames.py Raj magar 1234** Output: Names: ['Raj', 'magar', '1234']

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

if len(sys.argv[1:]) > 2:
    print("Names:", sys.argv[1:])
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:

A symbol table stores information about variables, functions, objects, and other entities in a program, including names, types, memory locations, and scoping information.

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

```
from math import *
```

Answer:

This is not recommended however, since there is high chance that conflicts between the imported names and existing variable names will occur.

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

Answer:

“dir()” function

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

Answer:

sys.path variable in Python is a list of directory locations used for searches to find modules and packages by interpreter when import statements are encountered in a script.

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

Answer:

While a Python program being executed as a script, the string value '__main__' will be assigned to the special variable __name__

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

Answer:

While Python program being imported as a module, the name of the module (i.e., the name of the script without the file extension) will be assigned to the special variable __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:

Because it allows for controlled execution flow and avoid accidental code execution on import.

Exercises are complete

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