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:

suffix.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 to use a special Integrated Development Environment (IDE) to write Python code in text files.

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, the results of evaluating expressions are not automatically displayed on the screen without the need of a `print()` function call.

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
import sys
print("Command line arguments:")
for arg in sys.argv[1:]:
    print(arg)
```

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:

String is the data-type of the `sys.argv` variable.

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

Answer:

sys.argv[0]

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:

```
PS C:\Users\Anil\OneDrive\Desktop> python PrintName.py
Command-line arguments:
PS C:\Users\Anil\OneDrive\Desktop> python PrintName.py arg1 arg2 arg3
Command-line arguments:
arg1
arg2
arg3
```

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) < 2:
    print("No names provided")
else:
    print("Command-line arguments:")
    for arg in sys.argv[1:]:
        print(arg)
```

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:

It stores informations about the scope and binding of names instances of various entities such as variable and function name, classes, objects, etc.

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

```
from math import *
```

Answer:

Because it imports all the names defined in the math module into the current namespace.

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

Answer:

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

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

Answer:

"__main__"value is assigned to the special variable `__name__`.

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

Answer:

The name of the module as a string is assigned to the `__name__` variable when a program has been 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:

Because it allows program to behave differently in each case.

Exercises are complete

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