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:

Script.py

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

Answer:

no

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

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:

Import sys module

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:

It stores the name of python script that was executed.

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

# Check if any command line arguments are passed
if len(sys.argv) > 1:
    # Print each command line argument passed
    for arg in sys.argv[1:]:  # Skip the first argument which is the script name
        print(arg)
else:
    print("No command line arguments passed.")
```

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

# Check if at least one name was passed as a command line argument
if len(sys.argv) > 1:
    # Print each name passed as a command line argument (skipping the script name)
    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:

Variable name , function name, class name, data type,memory address

---

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

```
from math import *
```

Answer:

it can lead to namespace pollution, conflicts, decreased readability, unintended imports, and potential performance issues.

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

Answer:

In interactive mode, the dir() function is the most convenient function to list all names (variables, functions, classes, etc.) defined within a module.

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

Answer:

The sys.path variable in Python is a list of directories that Python uses to search for modules and packages when you attempt to import them.

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

Answer:

assigned the value “__main__”

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

Answer:

Name of the 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 for a program to be able to detect whether it is running as a script or has been imported as a module because it allows the program to behave differently depending on how it is executed. This distinction provides flexibility in how the program can be reused and tested.

Exercises are complete

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