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.

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

©2021 Mark Dixon / Tony Jenkins

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** 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 to use a special Integrated Development Environment (IDE) to write

Python code in text files. You can use any text editor to write Python code, such as Notepad, Sublime Text, Atom, Vim, Emacs, and many more 12. However, using an IDE can make coding easier and more efficient by providing features such as syntax highlighting, code completion, debugging, version control, and more 12. Some popular Python IDEs include PyCharm, Spyder, Visual Studio Code, and Thonny

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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 when a script is executed from a file. You need to use the print() function to display the results 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:*

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 the program itself.

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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    def main(): args = sys.argv[1:]    for arg in args:  print(arg)    Output:  C:\Users\Sarita Gupta\Desktop>python PrintNames.py a b c a b c |

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    def main(): names = sys.argv[1:]    if len(names) == 0: print("No names provided") else: for name in names: print("Name provides is", name)    Output:  C:\Users\Sarita Gupta\Desktop>python PrintNames.py Sarita Name provides is Sarita    C:\Users\Sarita Gupta\Desktop>python PrintNames.py  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\_\_’ (filename)

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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 specially when run independently, not when imported as 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.