Functions

Exercises

Week 4

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.

What must be done before a function that is not *built-in* to Python can be used in a program?

Answer:

A custom function in Python has to be declared using the def keyword, called with the required arguments, handled for imports (if applicable), and have error handling implemented before it can be used.

_ Given the following `import` statement, how would a call to the `sin()` function be made? `import math`

Answer:

Math.sin() is used in the code to use the sin() function after the import statement import math.

_ Given the following `import` statement, how would a call to the `sqrt()` function be made? `from math import sqrt`

Answer:

Sqrt() is used in the code to call the function after the import statement from math import sqrt.

_ What is the name of the common library that is available with all Python distributions?

Answer:

The "Python Standard Library" is the common library that comes with every Python release.

_ What keyword is used in Python to define a new function?

Answer:

The keyword used in Python to define a new function is **“def”.**

Write some Python code that defines a function called `print_header(msg)`. This should output the value provided by the ‘`msg`’ parameter to the screen (prefixed by five asterisk ‘`*****`’) characters.

Answer:

```
def print_header(msg):
 print("*****", msg, "*****")
message = "Hello, this is a header"
print_header(message)
```

In the answer box below give an example of what the **docstring** may look like for the `print_header(msg)` function.

Answer:

```
def print_header(msg):
 """
 Print a header with five asterisks before and after the provided message.

 Parameters:
 - msg (str): The message to be displayed in the header.
 """
 print("*****", msg, "*****")
```

_ Where within a function definition should a **docstring** appear?

Answer:

The function body is preceded by a docstring and follows the function definition line (the line that contains the def keyword).

What statement should appear within a function's code block to cause a specific value to be passed back to the caller of the function?

Answer:

To force a certain value to be returned to the function's caller, the return statement must be included in the function's code block.

Write some Python code that defines a function called `find_min(a,b)` that returns the smallest of the two given parameter values.

Answer:

```
def find_min(a, b):
 """Return the smallest of two values."""
 return min(a, b)
input_a = float(input("Enter the first number: "))
input_b = float(input("Enter the second number: "))
result = find_min(input_a, input_b)
print(f"The smallest value between {input_a} and {input_b} is: {result}")
```

Given the following function definition, which of the *formal parameters* could be described as being a **default argument**?

```
def shouldContinue(prompt, answer=False):
      # function body...
```

Answer:

The formal parameter **'answer'** could be described as a default argument in the given function definition.

Provide two example calls to the above function, one which provides a value for the *default argument*, and one that does not.

Answer:

1)Providing a value for the default argument: **shouldContinue("Do you want to continue?", True)**
2) Not providing a value for the default argument: **shouldContinue("Do you want to continue?")**

_ State why following function definition would **not** be allowed.

```
def do_something(prefix="Message", prompt,
      answer=False): # function body...
```

Answer:

Because default arguments must come before non-default arguments when declaring functions in Python, the function definition is not permitted. This criterion is broken by the provided function, which takes a non-default argument (prompt) before a default parameter (prefix="Message").

What single character is placed directly before the name of a *formal parameter*, to indicate that a variable number of actual parameters can be passed when the function is called?

Answer:

The asterisk (*) is the single character that appears just before the name of a formal parameter to indicate that the function can receive a variable number of actual parameters when it is called.

What commonly used built-in function, which displays output on the screen, can take a **variable number** of arguments?

Answer:

'print()' is a frequently used built-in function that can accept a variable number of arguments and shows output on the screen.

Is it valid for a function's parameter name to be prefixed by two asterisk characters '**' as shown below?

```
def send_output(**details):
      # function body...
```

Answer:

Yes, it is valid for a function's parameter name to be prefixed by two asterisk characters ('**').

If present, what does this prefix indicate?

Answer:

It shows that a configurable number of keyword arguments can be passed to the parameter.

What is the name given to a small 'anonymous' function that must be defined using a single expression?

Answer:

"Lambda function" is the name given to a short 'anonymous' function that needs to be defined with only one expression.

Give an example of such a function that calculates the *cube* of a given number (i.e. the value of the number raised to the power of three) -

Answer:

```
number = float(input("Enter a number: "))
cube = lambda x: x ** 3
result = cube(number)
print(f"The cube of {number} is: {result}")
```

Exercises are complete

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