Done

Further Python

Introduction

The purpose of this practical is to continue using Python to develop programs. Don't forget that you are required to submit the indicated exercise through myLearn for assessment.

Python

We are now building up on our Python knowledge, and understand concepts such as iteration and functions. Iteration allows us to repeat a set of actions either for a fixed number of times (e.g. for each item in a list), or until a particular condition is met. Whereas functions allow us to group a set of actions together and "call" them with a single line of code - this makes it easier to break down larger problems into smaller ones, makes the code more readable and stops us having the same code in multiple places throughout our program (as a general rule of thumb, if you're writing code to perform the same task more than twice, you should move it into a function instead).

Exercises

Use material from this week's lectures to perform the following exercises:

1. **[This exercise should be submitted through the Assessments section of myLearn for Tutorial Exercise 3]** Write a function named *right_justify* that takes a string named *input_string* as a parameter. If the input string is longer than 79 characters, the string should be printed as is, Otherwise, the function should print the string with enough leading spaces that the last letter of the string is in column 80 of the display. (Note: Python has an inbuilt function called *len* that returns the length of a string. You can use this function to help perform the task).
2. Write a function named *draw_grid* that takes an integer named *width* and an integer named *height* as its parameters. The program should draw a grid like the following, with *width* -s across and *height* |s down. For example, *draw_grid(3,4)* should print:

```
+---+
|   |
|   |
|   |
|   |
+---+
```

3. Draw a stack diagram for the following program. What does the program print?

```
def b(z):
  prod = a(z, z)
  print(z, prod)
  return prod

def a(x, y):
  x = x + 1
  return x * y

def c(x, y, z):
  total = x + y + z
  square = b(total) ** 2
  return square

x = 1
y = x + 1
print(c(x, y + 3, x + y))
```

4. Using a while loop, write a function called *factorial* that takes a single parameter *n* and calculates *n!*(Note: *n! = n * (n-1) * (n-2) * ... * 1*)
5. Write a function called *has_duplicates* that takes a list as its only parameter and returns *True* if there is any element that appears in the list more than once. Otherwise the function should return *False*. Your function should not modify the original list.
6. The Python project hosts a language reference at https://docs.python.org/3/reference/index.html. Browse through the reference, especially Sections 6, 7, and 8 which detail the parts of Python we have already covered. This documentation could be useful for future reference.

Last modified: Thursday, 1 February 2024, 10:52 AM