Further Python

Exercise Answers

1. **[This exercise should be submitted through the Assessments section of Moodle 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:

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

```
def draw_grid(width, height):
    """Draw a grid with the given width and height.

      Arguments:
      width -- the width of the grid
      height -- the height of the grid
    """

    # end_line is the line to display at the top and bottom of the grid
    end_line = "+"
    # internal_line is the line to display for inside the grid
    internal_line = "|"

    # set up end_line and internal_line
    for i in range(width):
        end_line = end_line + "-"
        internal_line = internal_line + " "
    end_line = end_line + "+"
    internal_line = internal_line + "|"

    # print out the required grid
    print(end_line)

    for i in range(height):
        print(internal_line)

    print(end_line)

draw_grid(3,4)
```

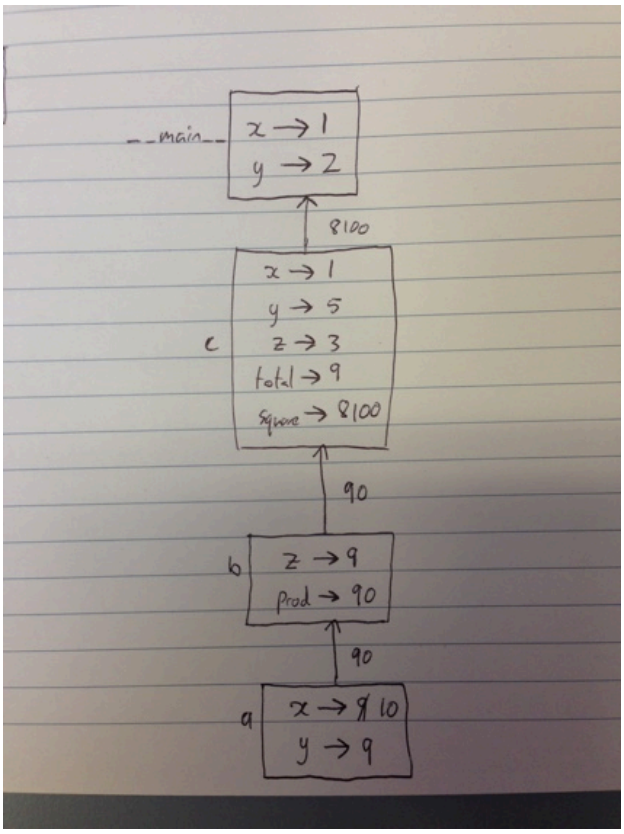
3. Draw a stack diagram for the following program. What does the program print?

```
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*)

```
def factorial(n):
    """Calculates the factorial of the given parameter."""
    product = 1
    while n > 0:
        product = product * n
        n = n - 1
    return product
```

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.

```
def has_duplicates(l):
    """Check if the given list has any duplicate items."""
    # The list of items that have been seen already (initially empty)
    seen = []
    for item in l:
        # If the item has already been seen, it must be a duplicate.
        # Return True immediately (no need to search rest of list)
        if item in seen:
            return True
        # The item has not been seen before
        # add it to the list of items that have now been seen
        seen.append(item)
    # Made it through the entire list - mustn't be duplicates
    return False
```

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, 10:44 AM