Practice problems

ISYS90088 Introduction to Application Development

Semester 2, 2016

-- adapted from dept. resources --

Exercises

1. Given the assignment `d = {"R": 0, "G": 255, "B": 0, "other": {"opacity": 0.6}}`, evaluate the following expressions, and determine: (a) the value the expression evaluates to; and (b) the final value of `d`. Assume that `d` is reset to its original value for each sub-question:

   (a) `d["R"]`

   (b) `d.pop("R")`

   (c) `d["R"] = 255`

   (d) `d["H"]`

   (e) `d.keys()`

   (f) `d["other"]["blur"] = 0.1`

   (g) `d[["H", "S", "L"]] = [120, 98, 5]`

   (h) `d["R", "B", "G"]`

2. What is the output of the following code:

```
def foo(x, y):

    print(x**y)

exp = foo(2,2)
print(exp)
```

3. What is the output of the following code:

```
def mutate(x, y):
    x = x + "--The End--"
    y.append("The End")
    print(x)
    print(y)

mystr = "It was a dark and stormy night."
mylist = mystr.split()
mylist2 = mylist
mutate(mystr, mylist2)
print(mystr)
print(mylist)
```

Problems

1. Write a function `freq_letter(string)` that takes a single argument `string` (a string) and returns a 2-tuple containing the most common letter in `string`, and how many times it occurs. In case of a tie, it should return the letter that occurred first in the string. For example:

```
>>> freq_letter('aardvark')
('a', 3)
>>> freq_letter('wooloomooloo')
('o', 8)
>>> freq_letter('abacuses')
('a', 2)
```

2. Write a program that prints the keys of a dictionary in descending order of their values. For example, for a dictionary `fruit_prices = {"apple": 0.5, "banana": 19, "durian": 7}`, your program should print:

```
banana
durian
apple
```

3. Write a function `make_catalogue` that creates a "catalogue" from a list of items, based on the following inputs: (1) the argument `items`, a list of strings (each naming an item to be inserted into the catalogue); and (2) an optional second argument `old_cat`, a set of unique strings (each naming an item in the existing catalogue). The function should return the set of unique items in the combination of `items` and (if provided) `old_cat`, and update `old_cat` in the process. For example:

```
>>> make_catalogue(['spam', 'spam', 'eggs'])
{'spam', 'eggs'}
>>> make_catalogue(['spam', 'spam', 'eggs'], old_cat={'eggs', 'milk'})
{'milk', 'eggs', 'spam'}
>>> my_cat = {'spam', 'socks'}
>>> make_catalogue(old_cat=my_cat, items=['socks', 'toothbrush'])
{'toothbrush', 'spam', 'socks'}
>>> my_cat
{'toothbrush', 'spam', 'socks'}
```