COMP10001 Foundations of Computing
Semester 2, 2022

Tutorial Questions: Week 6

— VERSION: 1477, DATE: AUGUST 29, 2022 —

Before the discussion questions, try Exercises 1–2 to revise last week's material

Discussion

1. In what situations would we use a "dictionary". How is it structured, how do we add and delete items?
2. What is the difference between using the `.pop()` method on a dictionary and using it on a list?
3. In what situations would we use a "set"? How does it differ from other "containers" such as lists and dictionaries?
4. What special operations can we perform on sets? How do we add and remove items from them?

   Now try Exercises 3–4

5. What is `None`? How is it used?
6. What is the difference between `sorted()` and `.sort()` when applied to a list? What does it mean to edit an object "in-place"?

Exercises

1. Do the following code snippets do the same thing? What are some advantages and disadvantages of each snippet? What if we needed a hundred different types of tool?

```
print("We␣need␣some␣saws")
print("We␣need␣some␣hammers")
print("We␣need␣some␣nails")
```

```
def get_str(part):
    return f"We␣need␣some␣{part}"

print(get_str("saws"))
print(get_str("hammers"))
print(get_str("nails"))
```

```
def get_str(part):
    return f"We␣need␣some␣{part}"

parts = ("saws", "hammers", "nails")

for part in parts:
    print(get_str(part))
```

2. Consider the following `while` loop and two conversions to `for` loops. Are the two `for` loops equivalent? Why might you choose one over the other?

```
count = 0
items = ('eggs', 'spam', 'more␣eggs')
while count < len(items):
    print(f"need␣to␣buy␣{items[count]}")
    count += 1
```

```
items = ('eggs', 'spam', 'more␣eggs')
for count in range(len(items)):
  print(f"need␣to␣buy␣{items[count]}")
```

```
items = ('eggs', 'spam', 'more␣eggs')
for item in items:
  print(f"need␣to␣buy␣{item}")
```

3. Evaluate the following given the assignment `d = {"R": 0, "G": 255, "B": 0, "other": {"opacity": 0.6}}`. If `d` changes as a result, give its new value. Assume `d` is reset to its original value each time.

   (a) `"R" in d`
   (b) `d["R"]`
   (c) `d["R"] = 255`
   (d) `d["A"]`
   (e) `d["A"] = 50`
   (f) `d.pop("G")`
   (g) `d["other"]["blur"] = 0.1`
   (h) `d.items()`

4. Evaluate the following given the assignment `s1 = {1, 2, 4}` and `s2 = {3, 4, 5}`. If `s1` or `s2` change as a result, give their new value. Assume `s1` and `s2` are reset to their original values each time.

   (a) `s1.add(7)`
   (b) `s1.add(2)`
   (c) `s2.remove(5)`
   (d) `s1 & s2`
   (e) `s1.union(s2)`
   (f) `s1 - s2`

Problems

1. Write a function which takes a string as input and prints the frequency of each character in the string using a dictionary. `freq_counts('booboo')` should print:

```
b 2

o 4
```

2. Write a function which takes two lists as input and returns a list containing the numbers which they both have in common. `in_common([1, 2, 4], [3, 4, 5])` should return `[4]`.

3. Write a function which takes a dictionary and returns a sorted list containing the unique values in that dictionary. `unique_values({'a': 1, 'b': 0, 'c': 0})` should return `[0, 1]`.

4. Write a function which takes a string, a character and an integer threshold and returns `True` if the character appears in the string with a frequency above the threshold, `False` if it appears at or below the threshold, and `None` if it doesn't appear at all. `above_thresh('I␣like␣the␣letter␣e', 'e', 3)` should return `True`.