COMP10001 Foundations of Computing
Semester 2, 2022

Tutorial Questions: Week 10

— VERSION: 1478, DATE: OCTOBER 3, 2022 —

Discussion

1. What is an "iterator"? What are some helpful methods in the `itertools` library?

   Now try Exercises 1

2. What is "recursion"? What makes a function recursive?

3. What are the two parts of a recursive function?

4. In what cases is recursion useful? Where should it be used with caution?

   Now try Exercise 2

Exercises

1. What output does the following code print?

```
import itertools
beatboxer = itertools.cycle(['boots', 'and', 'cats', 'and'])

for count in range(39):
    print(next(beatboxer))
```

2. Study the following mysterious functions. For each one, answer the following questions:

   - Which part is the base case?
   - Which part is the recursive case?
   - What does the function do?

3.
```
def mystery(x):
    if len(x) == 1:
        return x[0]
    else:
        y = mystery(x[1:])
        if x[0] > y:
            return x[0]
        else:
            return y
```

4.
```
def mistero(x):
    a = len(x)
    if a == 1:
        return x[0]
    else:
        y = mistero(x[a//2:])
        z = mistero(x[:a//2])
        if z > y:
            return z
        else:
            return y
```

Project 2

Today we'll be looking at two key functions in project 2: `proliferate` and `breed`. Let's start with a population of two wugs:

```
characteristics = ["intelligence", "beauty", "strength", "speed"]
superwug_genome = [1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
gene_zones      = [2, 1, 2, 3, 3, 1, 3, 3, 0, 0, 2, 2, 0, 1, 0, 1]

wug1           = ([1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0], 'F')
wug2           = ([1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1], 'M')
population = [wug1, wug2]
```

1. What superior characteristics do these wugs have? What `rank` would we assign to each wug?
2. The first step in `proliferate` is to work out how to clone each wug. What would the 16 clones of `wug1` look like? What are their ranks?
3. Now we call `proliferate` and observe the resulting population. Can you see which clones were added to the population?

   ```
   proliferate(population, limit=5)
   for wug in population:
       print(wug, rank(wug))

   >>> ([1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0], 'F') 4
   >>> ([1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0], 'M') 4
   >>> ([1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1], 'M') 3
   >>> ([0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0], 'F') 3
   >>> ([1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0], 'F') 3
   ```

4. The first step in the `breed` function is to score the suitability of mates. From the perspective of `wug1`, the suitability of `wug2` is calculated as `wug2`'s rank plus a `coincidence_bonus` times the number of features they share. Using `coincidence_bonus = -2`, what would this score be?

Problems

1. Write a recursive function which takes an integer `n` and calculates the n^{th} fibonacci number. The 0^{th} fibonacci number is 0, the 1^{st} fibonacci number is 1 and all following fibonacci numbers are defined as the sum of the preceding two fibonacci numbers. `fib(10)` should return `55`
2. Write a function which takes two strings as input and uses an `itertools` iterator to find whether the first word is an anagram of the second word. This might not be a very efficient way to find an anagram but it will help us work with iterators! `anagram('astronomer', 'moonstarer')` should return `True`
3. Write a function which takes a lowercase string as input and prints the frequency of each vowel in the string using a dictionary. `vowel_counts('i␣love␣python')` should print:

   ```
   i 1
   e 1
   o 2
   ```

4. Write a function which takes two lists of integers and returns the average of the numbers which they both have in common. `in_common_average([1, 2, 3, 4, 5], [0, 2, 4, 6])` should return `3.0`