Computer Science 125
24 September 2020

1. What is produced by the following code?

(a)

```
[n*n for n in range(10)]
```

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

(b)

```
[n*n for n in range(10) if n % 3 == 1]
```

expression

[1, 16, 49]

n is 1 more than a multiple of 3

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

squared

(c)

```
[c*2 for c in "banana"]
```

["bb", "aa", "nn", "aa", "nn", "aa"]

c = "b" c*2 → "bb"

(d)

```
text = "the quick brown fox jumps over the lazy dog"
fun = [len(w) for w in text.split() if w != "the"]
print(fun)
```

5 5 3 5 4 4 3

[5, 5, 3, 5, 4, 4, 3]

(e)

```
words = ["crew", "brick", "swing", "cat"]
puzzle = "".join(words)
print(puzzle)
```

"crewbrickswingcat"

```
words = ["crew", "brick", "swing", "cat"]
puzzle = " ".join(words)
print(puzzle)
```

space character

"crew brick swing cat"

2. After the following code is run, what will be stored in `nums` and in `vals`?

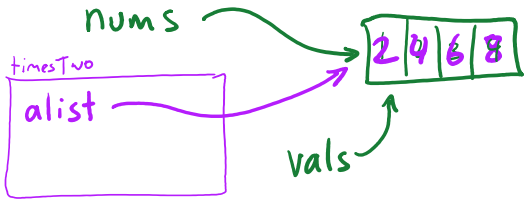
```
def timesTwo(alist):
    for i in range(len(alist)):
        alist[i] = 2*alist[i]
    return alist

nums = [1, 2, 3, 4]
vals = timesTwo(nums)
```

← changes the values in the list nums

nums = [1, 2, 3, 4]

vals = [2, 4, 6, 8]



How could you modify the previous function so that it doesn't modify the contents of `nums`?

1

```
def timesTwo(alist):
    newlist = alist[:]
    for i in range(len(newlist)):
        newlist[i] = 2*newlist[i]
    return newlist

nums = [1, 2, 3, 4]
vals = timesTwo(nums)
```

2

```
def timesTwo(alist):
    return [2*i for i in alist]

nums = [1, 2, 3, 4]
vals = timesTwo(nums)
```

3

```
def timesTwo(alist):
    newlist = []
    for i in range(len(alist)):
        newlist.append(2*alist[i])
    return newlist

nums = [1, 2, 3, 4]
vals = timesTwo(nums)
```

3. In Python, how is a tuple different from a list?

tuples: not mutable

(,)

lists: mutable

[,]

What is printed by the following code?

```
def fun(text):
    length = len(text)
    first = text[0]
    return length, first

print(fun("computer"))
```

← same as `return (length, first)`

(8, c)

Letter Frequencies Problem:

```
def letterFrequencies(text)
    counts = [0] * 26      # set up a list of 26 accumulators
    for c in text:
        i = ord(c.lower()) - 97
        counts[i] += 1
    return counts
```

} increment the appropriate counter

Mean and Standard Deviation problem:

```
def meanSD(nums):
    n = len(nums)
    mean = sum(nums)/n
    diffs = [(x-mean)**2 for x in nums]
    s = math.sqrt(sum(diffs)/(n-1))
    return mean, sd
```

$\sum (x_i - \bar{x})^2$ (value, mean)

MORE PRACTICE WITH LISTS – SOLUTIONS

CS 125

Working with a partner/group, use the following steps to solve each of the following problems.

(a) Plan your code on the white board (either on the classroom wall or on Zoom). Write out your entire program. Think about what errors might occur and how to fix them.

(b) Plan multiple test cases. What input will you send to your function? For each input, what value should be returned?

(c) *Only after you have completed steps (a) and (b) should you type your code in Python.*

(d) After you have typed your code, run your test cases. Does your code work? If not, how can you fix it?

1. Write a list comprehension to solve each of the following problems. (Your solution to each of the following problems should be a single line of code.)

   a. Make a list of all integers between 1 and 1000 that are divisible by 11.

   ```
   [i for i in range(1,1001) if i % 11 == 0]
   ```

   b. Remove all of the vowels from a string.

   ```
   "".join([c for c in "this is a test string" if c not in "aeiouAEIOU"])
   ```

   c. Make a list of all integers between 1 and 1000 that include the digit 5.

   ```
   [i for i in range(1,1001) if "5" in str(i)]
   ```

2. Write a function `letterFrequencies(text)` that counts the number of times each letter occurs in a string. Your function should return a list of 26 counts, one count for each letter of the alphabet.

   For example: `letterFrequencies("Hat Cap Sat")` returns
   `[3,0,1,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,1,2,0,0,0,0,0,0]`

   Hint: one way to convert an uppercase or lowercase character `c` to its position in the alphabet is:

   ```
   ord(c.lower()) - 97
   ```

   ```
   import string
   def letterFrequencies(text):
       counts = [0]*26
       for c in text:
           if c in string.ascii_letters:
               i = ord(c.lower()) - 97
               counts[i] += 1
       return counts
   ```

3. Write a function `meanSD(nums)` that accepts a list of numbers and returns a tuple containing the mean and standard deviation of the numbers. If the numbers are $x_1, x_2, \ldots, x_N$, then the mean and standard deviation are computed as follows:

   - mean: $\bar{x} = \frac{x_1 + x_2 + \cdots + x_N}{N}$ (this is the average of the numbers)
   - standard deviation: $s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$

   For example, `meanSD([2, 5, 7, 8, 12, 15])` returns `(8.16667, 4.70815)`.

```
import math
def meanSD(nums):
    n = len(nums)
    mean = sum(nums)/n

    diff2 = 0
    for x in nums:
        diff2 += (x - mean)**2
    sd = math.sqrt(diff2/(n-1))

    return (mean, sd)

print(meanSD([2,5,7,8,12,15]))
print(meanSD([4,4,4,4]))
```

4. A string is said to be *complete* if it contains all of the letters from *a* to *z*. Write a function `isComplete(text)` that accepts a string of text and determines whether the string is complete. If the string is complete, the function returns `True`, otherwise it returns `False`. As part of your solution, you may wish to call your `letterFrequencies` function from #2.

```
def isComplete(text):
    counts = letterFrequencies(text)
    complete = 0 not in counts # evaluates to True or False
    return complete

print(isComplete("The quick brown fox jumps over the lazy dog.")) #
True
print(isComplete("a string")) # False
```