Lec08-strings-RyanSponzilli

September 19, 2024

1 ASTR 310 Lecture 8

1.0.1 1. List comprehensions, mapping, formatted output

Using list comprehensions and/or mapping and/or zip [Reading 7], write Python code to:

- Create a list x containing 10 values equally spaced from 0 to 2π inclusive.
- Create a second list y containing values of $x \sin^2(3x/\pi)$.
- Create a third list z that is True if $y > 1.5$ and False otherwise.
- Print a table of values containing x and y, formatted in columns 10 characters wide with 5 decimal places, and z, formatted in columns 5 characters wide. You can use either a string or an integer format for the booleans.

We're still avoiding numpy arrays! You must continue to use lists.

[7 pts total]

[81]:
```
from math import *

x = [0,]
for i in range(9):
    x.append(x[-1] + 2*pi/9)

y = [i * sin(3*i/pi) ** 2 for i in x]

z = [i > 1.5 for i in y]

table = ["{x:10.5f}{y:10.5f}{z:5}".format(x=x[i],y=y[i],z=z[i]) for i in␣
 ↪range(10)]

for item in table:
    print(item)
```

```
   0.00000   0.00000    0
   0.69813   0.26695    0
   1.39626   1.31900    0
   2.09440   1.73169    1
   2.79253   0.58391    0
   3.49066   0.12677    0
   4.18879   2.39913    1
```

```
4.88692   4.87671    1
5.58505   3.69454    1
6.28319   0.49055    0
```

1.0.2 2. Working with a file

- Write the output from exercise 1 to a text file called "table.out".
- Read this file back in to three lists a, b, c, and verify that they are the same as x, y, z.

The verification step can be surprisingly difficult due to the formatted printing. If you are new to coding you can just print the lists and verify by eye that they are the same. If that is way too easy for you, then you can write some more detailed code to check.

[5 pts total]

[82]:
```
f = open("table.out", 'w')
f.writelines(map(lambda x: x + "\n", table))
f.close()

f = open("table.out", 'r')
check = f.readlines()
f.close()
a = [float(check[i][0:10]) for i in range(10)]
b = [float(check[i][10:20]) for i in range(10)]
c = [float(check[i][20:25]) for i in range(10)]
a,b,c
```

[82]:
```
([0.0,
  0.69813,
  1.39626,
  2.0944,
  2.79253,
  3.49066,
  4.18879,
  4.88692,
  5.58505,
  6.28319],
 [0.0,
  0.26695,
  1.319,
  1.73169,
  0.58391,
  0.12677,
  2.39913,
  4.87671,
  3.69454,
  0.49055],
 [0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0])
```

1.0.3 3. String mayhem

- Get a string from the user and count the number of times each vowel (a, e, i, o, u) is encountered. Print the counts in a nicely formatted table.
- Replace any number words (six, seventeen) with the corresponding integer (up to nineteen) and print the resulting string. Hints: you could construct a dictionary, or you could construct a list of number words in which the list index of each number word is its corresponding value. Also, strings behave like lists of characters so you could consider the list operations described in Reading 7.

For example:

Enter string: There are seventeen cats in twelve buildings with four bowls in each.
a: 3
e: 10
i: 5
o: 2
u: 2
There are 17 cats in 12 buildings with 4 bowls in each.

[8 pts total]

[83]:
```
str_input = input()

count_a = str_input.count('a')
count_e = str_input.count('e')
count_i = str_input.count('i')
count_o = str_input.count('o')
count_u = str_input.count('u')

counts = {'a': count_a, 'e': count_e, 'i' : count_i, 'o' : count_o, 'u' :␣
 ↪count_u}

nums = ["zero ", "one ", "two ", "three ", "four ", "five ", "six ", "seven ",␣
 ↪"eight ", "nine ", "ten ", "eleven ", "twelve ", "thirteen ", "fourteen ",␣
 ↪"fifteen ", "sixteen ", "seventeen ", "eighteen ", "nineteen "]

for i in range(len(nums)):
    str_input = str_input.replace(nums[i], str(i) + " ")

for key in counts:
    print("{k:1}: {n:3}".format(k=key,n=counts[key]))

print(str_input)
```

```
a:   3
e:  10
i:   5
o:   2
```

```
u:   2
There are 17 cats in 12 buildings with 4 bowls in each.
```