# Section Handout #8

1. List Mystery

Consider the following function:

def list\_mystery(a):

for i in range(1, len(a)):

a[i] = i + a[i - 1] - a[i]

Indicate in the right-hand column what values would be stored in the list after the function list\_mystery executes if the list in the left-hand column is passed as a parameter to it.

|  |  |
| --- | --- |
| Original Contents of List | Final Contents of List |
| a1 = [7]  list\_mystery(a1)  a2 = [4, 3, 6]  list\_mystery(a2)  a3 = [7, 4, 8, 6, 2]  list\_mystery(a3)  a4 = [10, 2, 5, 10]  list\_mystery(a4)  a5 = [2, 4, -1, 6, -2, 8]  list\_mystery(a5) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

2. While Loop Simulation

For each call below to the following method, write the value that is returned:

def mystery(x):

a = 1

c = 0

while x > 0:

a = x % 2

if a == 1:

c += 1

x = x // 2

return c

|  |  |
| --- | --- |
| Method Call | Value Returned |
| mystery(2)  mystery(-1)  mystery(7)  mystery(18)  mystery(43) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

**3. Assertions**For the following function, identify each of the three assertions in the table below as being either ALWAYS true, NEVER true or SOMETIMES true / sometimes false at each labeled point in the code. (You may abbreviate these choices as A/N/S respectively.)

def mystery(x):

y = 1

z = 0

***# Point A***

while y <= x:

***# Point B***

y = y \* 10

z += 1

***# Point C***

***# Point D***

Z -= 1

***# Point E***

return z

|  |  |  |  |
| --- | --- | --- | --- |
|  | y > x | z < 0 | z > 0 |
| Point A |  |  |  |
| Point B |  |  |  |
| Point C |  |  |  |
| Point D |  |  |  |
| Point E |  |  |  |

4. File Processing

Write a function named print\_strings that takes file name as a parameter. The file contains pairs containing a number first and a String second. Your code should read the file and produce one line of output for each pair with the given String repeated the given number of times. For example if the file contains the following data:

6 fun. 3 hello 10 <> 2 25 4 wow!

3

t

your function should produce the following output:

fun.fun.fun.fun.fun.fun.

hellohellohello

<><><><><><><><><><>

2525

wow!wow!wow!wow!

ttt

Notice that there is one line of output for each integer/string pair. The first line has 6 occurrences of "fun.", the second line has 3 occurrences of "hello", the third line has 10 occurrences of "<>", the fourth line has 2 occurrences of "25", the fifth line has 4 occurrences of "wow!" and the sixth has three occurrences of “t”. Notice that there are no extra spaces included in the output. You are to exactly reproduce the format of this sample output. You may assume that the input values always come in pairs with an integer followed by a String (which itself could be numeric, such as "25" above). If the file is empty (no integer/string pairs), your function should produce no output.

You may not use string multiplication.

5. Programming

Write a function called process\_scores that takes the name of a file as a parameter. The file contains a series of lines that represent student records. Each student record takes up two lines of input. The first line has the student's name and the second line has a series of plus and minus characters. Below is a sample input:

Kane, Erica L.

--+-+

Chandler, Adam

++-+

Martin, Jake Daniel

+++++++

Dillon, Amanda

++-++-+-

The number of plus/minus characters will vary, but you may assume that at least one such character appears and that no other characters appear on the second line of each pair.

For each student you should produce a line of output with the student's name followed by a colon followed by the percent of plus characters. For example, if the input above is stored in a file called input.txt and we make the following call:

process\_scores("input.txt")

The following output should be produced:

Kane, Erica L.: 40.0% plus

Chandler, Adam: 75.0% plus

Martin, Jake Daniel: 100.0% plus

Dillon, Amanda: 62.5% plus

You must exactly reproduce this format.

6. Programming

Write a function named longest\_name that reads names typed by the user and prints the longest name (the name that contains the most characters) in the format shown below. Your method should accept an integer n as a parameter and should then prompt for n names.

The longest name should be printed with its first letter capitalized and all subsequent letters in lowercase, regardless of the capitalization the user used when typing in the name.

If there is a tie for longest between two or more names, use the tied name that was typed earliest. Also print a message saying that there was a tie, as in the right log below. It's possible that some shorter names will tie in length, such as ryan and TITO in the left log below; but don't print a message unless the tie is between the longest names.

You may assume that n is at least 1, that each name is at least 1 character long, and that the user will type single-word names consisting of only letters. The following table shows two sample calls and their output.

|  |  |  |
| --- | --- | --- |
| **Call** | longest\_name(5) | longest\_name(7) |
| **Output** | name #1? **ryan**  name #2? **TITO**  name #3? **John**  name #4? **lAuRaLyN**  name #5? **SujaN**  Lauralyn's name is longest | name #1? **PeTer**  name #2? **eric**  name #3? **RAFAEL**  name #4? **brian**  name #5? **sarina**  name #6? **LIOR**  name #7? **EmIlIo**  Rafael's name is longest  (There was a tie!) |

### Solutions

1.

|  |  |
| --- | --- |
| Call  a1 = [7]  list\_mystery(a1)  a2 = [4, 3, 6]  list\_mystery(a2)  a3 = [7, 4, 8, 6, 2]  list\_mystery(a3)  a4 = [10, 2, 5, 10]  list\_mystery(a4)  a5 = [2, 4, -1, 6, -2, 8]  list\_mystery(a5) | Final Contents of Array  [7]  [4, 2, -2]  [7, 4, -2, -5, -3]  [10, 9, 6, -1]  [2, -1, 2, -1, 5, 2] |

2.

14 14

7 9 14 1

18 18

7 9 14 18

3. Assertions

|  |  |  |  |
| --- | --- | --- | --- |
|  | y > x | z < 0 | z > 0 |
| Point A | SOMETIMES | NEVER | NEVER |
| Point B | NEVER | NEVER | SOMETIMES |
| Point C | SOMETIMES | NEVER | ALWAYS |
| Point D | ALWAYS | NEVER | SOMETIMES |
| Point E | ALWAYS | SOMETIMES | SOMETIMES |

4.

def print\_strings(file\_name):

with open(file\_name) as file:

text = file.read().split()

for i in range(0, len(text), 2):

times = int(text[i])

word = text[i + 1]

for i in range(0, times):

print(word, end=’’)

print()

5.

def process\_scores(file\_name):

with open(file\_name) as file:

lines = file.readlines()

for i in range(0, len(lines), 2):

name = lines[i]

data = lines[i + 1]

plus = 0

count = 0

for i in range(0, len(data)):

count += 1

if (data[i] == '+'):

plus += 1

percent = 100 \* plus / count

print(name + ": " + str(percent) + "% plus")

6.

def longest\_name(names):

longest = input("name #1? ")

count = 1

for i in range(2, names + 1):

name = input("name #" + i + "? ")

if (len(name) == len(longest)):

count += 1

if (len(name) > len(longest)):

longest = name

count = 1

fixed\_name = longest[0].upper()

fixed\_name = fixed\_name + longest[1:].lower()

print(fixed\_name + "'s name is longest")

if (count > 1):

print("(There was a tie!)")