CS 120: Intro to Computer Programming II

In-Class Activity - 01 Python Review - Day 2

Activity 1 - Turn in this one

Write a file named `ica_01_2_act1.py`

Read two lines of input from the user, and do something with them. Check the testcases to find out what.

HINT: The `range()` function in Python has an exclusive upper bound - meaning that you'll need to so something special to make this work ...

Solution:

```
beg = int(input())
end = int(input())

if beg > end:
    for i in range(beg, end-1, -1):
        print(i)
else:
    for i in range(beg, end+1):
        print(i)
```

Activity 2 - Turn in this one

Write a file named `ica_01_2_act2.py`

NOTE 1:
This program uses random numbers. In order to test a program using random numbers, we need to make sure that your program always generates random numbers in a predictable pattern. To do this, we "seed" the RNG (random number generator) with a starting value, like this:

```
random.seed(some_integer)
```

NOTE 2:
To generate a random integer, call:

```
random.randint(MIN,MAX)
```

NOTE 3:
You will need to import the `random` library to use these functions.

Write a function `random_rolls(s)` which takes one parameter. Seed the random number generator with the value that you've been given; then enter a loop, which generates a series of random numbers. Each number generated should be in the range 1 to 20 (inclusive).

See the testcases to see what you will do with the random numbers, and also to find out when the loop will end.

Solution:

```
import random

def random_rolls(s):
    random.seed(s)

    total = 0
    while True:
        roll = random.randint(1,20)

        if roll == 1:
            print("Rolled a 1, the loop will now terminate.")
            break
        elif roll == 20:
            print("Rolled a 20, I will add double the value to the running sum.")
            total += 40
        else:
            print(f"Rolled a {roll}, I will add the value to the running sum.")
            total += roll

        print(f"  Sum so far: {total}")

    return total
```

Activity 3 - Turn in this one

Write a file named `ica_01_2_act3.py`

Write a function `count_chars(text,c)` which takes two arguments: a string, and a single character.

What will you use this dictionary for? See the testcases to find out!

Instructor's Note:
I noticed, after composing this problem, that it was possible to complete it without a dictionary. But I want you to practice with dictionaries, so please use one, even if it's not necessary.

Solution:

```
def count_chars(text,c):
    counts = {}

    for c in msg:
        if c not in counts:
            counts[c] = 1
        else:
            counts[c] += 1

    print("Passing the set of keys from the dictionary to the sorted() function...")
```

```
        print("Printing out the dictionary contents, in character order:")
        for k in sorted(counts):
            print(f"{k} : {counts[k]")

        print("Calculating the total count of all characters <= to the argument...")
        total = 0
        for k in counts:
            if k <= c:
                total += counts[k]
        return total
```

(activity continues on next page)

Challenge Activity - Do not turn in this one

This activity is about coming up with an **explanation,** not code.

First, experiment with what happens when you convert a string, directly into an array. For example, write code something like this:

```
msg = "This is an interesting message"
x   = list(msg)
```

what is the value of `x` at the end of this code? Explain this in the writeup that you turn in.

Now, write a function which does the same thing as the `list()` constructor above - but you must do it with a **loop;** don't convert the string to an array directly. Return the resulting array.

Solution: When you convert a string into an array, it builds an array where every character in the string is one element in the array.

```
def to_list(data):
    retval = []
    for elem in data:
        retval.append(elem)
    return retval
```

Challenge Activity - Do not turn in this one

Write a function which takes two parameters: an array of values, and a single value to compare them with. (You **must not** modify either parameter.) Build two new arrays, filling them up with values from the parameter; the first array must contain only values **less than** the comparison value, and the second must contain only values which are greater than or equal to it.

However, the values in both of the new arrays must be in the same order as they were in the original array (although, obviously, with some things missing). So your best strategy will be to iterate through the original array, considering each value in turn, and placing it into the appropriate destination array.

Print out both arrays.

EXAMPLE

```
input:     [74, -4, -37, 36, 54, 57, -13, 35, -2, 4]
compare:   50

output 1:  [-4, -37, 36, -13, 35, -2, -4]
output 2:  [74, 54, 57]
```

Solution:

```
def partition(vals, split_pt):
    lt = []
    ge = []

    for v in vals:
        if v < split_pt:
```

```
                lt.append(v)
            else:
                ge.append(v)

        print(lt)
        print(ge)
```