Review Problems - DS 2000/2001 F 2021

(Please note that these practice problems are ***not*** reflective of the format of the DS 2000 midterm. These are intended to be an extra tool for you to study the content of this course so far!)

Output and loops

Write the output of the following code snippets.

Code Snippet	Output

`my_fave_num = 97`
`if my_fave_num > 0:`
`    print("so big")`
`if my_fave_num < 100:`
`    print("so small")`	

`my_fave_num = 13`
`if my_fave_num > 0:`
`    print("so big")`
`elif my_fave_num < 100:`
`    print("so small")`	

`party = "birthday"`
`p = 0`
`while p < len(party):`
`    print(party[p] + party)`
`    p += 1`	

`target = ""`
`doodads = ["bauble", "cat", "lamp", "stop sign"]`
`ind = 0`
`while ind < len(doodads):`
`    print(str(len(target)) + ": " + doodads[ind])`
`    target = doodads[ind]`
`    ind += 1`	

`s = "skee ball"`
`num = 0`
`while num < len(s):`
`    print(s[len(s) - num - 1])`
`    num += 1`	

```
amount = 1
items = ["toothbrush", "toothpaste", "volcano"]
i = 0
while i < len(items):
    amount = amount + len(items[i])
    print(amount)
    i += 1
```

Code Snippet	Answers

```
x = 10
y = 2
while y < x:
     print(x)
     x += 10
```
	Number of times this loop executes:

```
x = 10
animals = ["cat", "dog", "bear"]
for animal in animals:
     print(x)
     x += 10
```
	Number of times this loop executes:

```
my_s = "ds2001"
index = 0
while index < len(my_s) - 2:
     index += 1
     print(my_s[index])
```
	The output of this code snippet:

```
my_s = "ds2001"
for i in range(len(my_s)):
     print(my_s[i])
```
	The output of this code snippet:

How many times does the *inner* loop iterate for the following code snippet?

```
count = 0
while count < 3:
    count2 = 0
    while count2 < 4:
        print(count2)
        count2 += 1
    count += 1
```

number of iterations of inner loop:

Challenge: how many times does the *inner* loop iterate for the following code snippet?

```
for outer in range(5):
    count = 0
    print("outer:", outer)
    while count < outer:
        print("count:", count)
        count += 1
```

number of iterations of inner loop:

Fill in the blanks so that the following code snippet computes what the comments indicate.

```
# create one string by duplicating all letters in a string.
# Example "resume101" => "rreessuummee110011"

target = ______________  # string to duplicate here (updated 3/20)

two_times = _____________

count = _________________

while ____________________________________:

    ___________ = _________________ + __________________

    count = _______________ + __________________


print(_____________)  # print the final string
```

Next, write the same code but use a for loop to do so instead!

Writing Functions

a) Write a function, `number_words`, that takes one string as a parameter and **returns** the number of words in that string. You may assume that all words are separated by spaces.

Function call	Return value
`number_words("Hello there")`	2
`number_words("Hello")`	1
`number_words("Hello there I am an ocelot")`	6
`number_words("Wow! I'm glad I'm studying for this midterm!")`	8
`number_words("I'm excited to use my awesome programming skills in other domains!")`	11

As a challenge, re-write this function, but *without* using the `str.count()` or `str.split()` methods!

b) Write a function, `sandwich`, that takes two strings as parameters, s1 and s2, and **prints** a new string composed of the first string repeated n times where n is the length of the first string, followed by the second string, followed by the first string n times.

Function call	Output
`sandwich("a","cheese")`	"acheesea"
`sandwich("bb","spinach")`	"bbbbspinachbbbb"
`sandwich("ham","cheese")`	"hamhamhamcheesehamhamham"

c) For the following function, select all types for the parameters that will cause it to run without errors, then describe what the function does.

code:	
	```
	def mystery3(param1, param2):
	    var1 = 0
	    while var1 < len(param1):
	        print(param1[var1])
	        var1 += param2
	```

Type of `param1` (circle all that will run without errors):	int float string boolean list of ints list of floats list of strings list of booleans
Type of `param2` (circle all that will run without errors):	int float string boolean list of ints list of floats list of strings list of booleans

A better name for this function:	
What does the function do:	

d) Write a function, `same_maximum`, that takes two parameters, lists or strings, and prints "different types!" if they are not either both strings or both lists, "same max: [the maximum]" if they have the same maximum element, and "different max!" if they don't. Use the `type()` and the `max()` functions to help you. You may assume that only strings or lists are passed to your function.

Example Function Calls	Output
`same_maximum(["a", "b"], "b")`	`different types!`
`same_maximum([7, 2], [1, -3, 7, 2, 4])`	`same max: 7`
`same_maximum([100, 2], [1, -3, 7, 2, 4])`	`different max!`
`same_maximum("zebra", "the letter z")`	`same max: z`

e) Write a function, `sum_positives`, that takes one list of numbers as a parameter and uses a loop to calculate then return the sum of all positive numbers from the list.

Example Function Calls	Return value
`sum_positives([1])`	`1`
`sum_positives([-2, 2, -2, 3])`	`5`
`sum_positives([-2, 0, -2, 0, -100])`	`0`
`sum_positives([])`	`0`

f) Write a function to extract course number (returned as an integer) from a course name (taken as a parameter) like “DS2001” gives 2001, “CS4120” gives 4120 and so on. You may assume that the department code is always 2 letters long.

Function call	Return value
`course_number("DS2000")`	`2000`
`course_number("DS2001")`	`2001`
`course_number("CS4120")`	`4120`

g) Write a function to display a table of dice rolls for a given list of people. Each person will roll a six-sided die 5 times. Return a list of lists where each row in the list represents the rolls that one person made.

Function call	Output (printed)

`roll_table(["Felix", "Arushi"])`	`Felix: 1 1 1 3 6`
	`Arushi: 5 3 2 1 3`

`roll_table(["Asa", "Smit", "Archit"])`	`Asa: 4 4 2 6 4`
	`Smit: 2 1 1 3 2`
	`Archit: 6 6 2 6 3`

h) Now, update your function to ***return*** a list of lists where each row in the list represents the rolls that one person made.

Function call	Return value
`roll_table(["Felix", "Arian"])`	`[[1, 1, 1, 3, 6], [5, 3, 2, 1, 3]]`

i) Write a function, `end_digits`, that takes two parameters, an integer representing a target number and an integer n, and **prints** the n ending digits of the target number. (Hint: think about using the % operator or about converting the integer to a string)

Function call	Output (printed)
`end_digits(123, 1)`	`3`
`end_digits(123, 2)`	`23`
`end_digits(123, 3)`	`123`