ITI 1120

Lab # 6

while loops, lists …

Anther very important lab. Please make sure you learn and understand all of this material.

Starting Lab 6

- Open a browser and log into Brightspace
- On the left hand side under Labs tab, find lab6 material contained in lab6-students.zip file
- Download that file to the Desktop and unzip it.

Before starting, always make sure you are running Python 3

This slide is applicable to all labs, exercises, assignments … etc

ALWAYS MAKE SURE FIRST that you are running Python 3.

That is, when you click on IDLE (or start python any other way) look at the first line that the Python shell displays. It should say Python 3.

If you do not know how to do this, read the material provided with Lab 1. It explains it step by step

Do all the exercises labeled as Task in your head i.e. on a paper

Later on if you wish, you can type them into a computer (or copy/paste from the solutions once I poste them)

Task 1

Go to the page below in your interactive textbook
-read through
-click Forward until reaching the end on the Python Visualizing example (to understand it)
-and answer the two questions and the end:

https://runestone.academy/ns/books/published/thinkcspy/MoreAboutIteration/ThewhileStatement.html

Study of while vs for … in some elementary functions:

Problem: sum of positive odd integers smaller than n

A solution with for loop

```
def sum_odd_for(n):
    odd_sum = 0
    for num in range(n):
        if num % 2 == 1:
            odd_sum = odd_sum + num
    return odd_sum
```

A solution with a while loop

```
def sum_odd_while(n):
    odd_sum = 0
    i=1
    while i < n:
        if i % 2 == 1:
            odd_sum = odd_sum + i
        i=i+1
    return odd_sum
```

Problem: Sum of odd numbers in a given list a

A solution with for loop

```
def sum_list_for(a):
    ''' (list)->num '''
    list_sum = 0
    for i in range(len(a)):
        if a[i] % 2 == 1:
            list_sum = list_sum +a[i]
    return list_sum
```

A solution with a while loop

```
def sum_list_while(a):
    ''' (list)->num
    list_sum = 0
    i=0
    while i < len(a):
        if a[i] % 2 == 1:
            list_sum = list_sum + a[i]
        i=i+1
    return list_sum
```

Study of while vs for … in another important elementary functions:

Problem:
find a minimum element in a list (of length at least 1)

Two solutions with for loop
(for loop over elements and over indices)

```
def minimum_for_v1(a):
    ''' (list)->num '''
    minimum = a[0]
    for element in a:
        if element < minimum:
            minimum = element
    return minimum
```

```
def minimum_for_v2(a):
    ''' (list)->num '''
    minimum = a[0]
    for i in range(len(a)):
        if a[i] < minimum:
            minimum = a[i]
    return minimum
```

A solution with a while loop

```
def minimum_while(a):
    ''' (list)->num'''
    minimum = a[0]
    i=0
    while i < len(a):
        if a[i] < minimum:
            minimum = a[i]
        i=i+1
    return minimum
```

Task 2 and Programming Exercise 1

After studying the previous four functions

1.Open the file called **seven_functions.py** Copy/paste, one by one, the three solutions labeled with "# with while loop" into Python visualizer. Run through each example and understand how the solutions work and how the variables change in the loops. Play by changing what variable i is initialized to and how it is incremented (do for example i=i+3) and see what happens. As always, you can find python visualizer here (make sure you choose Python 3)

http://www.pythontutor.com/visualize.html#mode=edit

2.Programming exercise 1: Open file called **prog-ex-1.py**
Complete the function sum_odd_while_v2 that takes an integer n as input parameter and computes the sum of all odd integers between 5 and n (inclusively). Use **WHILE LOOP only** and **NO IF STATEMENTS.**

```
>>> sum_odd_while_v2(10)
21
>>> sum_odd_while_v2(-7)
0
```

Programming Exercise 2

Write a program using while loop that

- asks the user for two integers (one by one or at the same time if you know how to handle it). Then it adds them and displays the sum.

Then the user is asked to enter 'yes' if she wishes to perform the operation again, and otherwise if she enters anything other than 'yes' the program terminates. The operation (described in the bullet above) is repeated, as long as the user enters 'yes'.

Task 3

Question 1. What does the program below print if the input user entered was:

1 5 -3 0

Question 2. Can you tell what the program does? Write one sentence explaining what the function does, in plain English.

```
number=int(input("Enter an integer: "))
apple=number

while(number!=0):
      number=int(input("Enter an integer: "))
      if(number > apple):
            apple=number
print("apple is", apple)
```

Task 4

Question 1. How many times does the program below print aha?

```
x = 10
y = 100 % 90
i= 0
while i < 10000:
      if(x!=y):
            print("aha")
      i=i+1
```

Task 5

Question 1. What does the the code on the right print?

ps. sometimes I say code, sometimes I say program. I mean the same thing.

```
def is_apple(x):
    if x%4 == 0:
        appleFlag = True
    else:
        appleFlag = False
    appleFlag = appleFlag and is_pear(-200)
    return appleFlag

def is_pear(x):
    if (x < -101):
        pearFlag = True
    else:
        pearFlag = False
    return pearFlag

flag1 = is_apple(44)
flag2 = is_pear(-101)
print(str(flag1)+" "+str(flag2))
if (flag1 and flag2):
    print("ding!")
if (flag1 or flag2):
    print("dong!")
```

Task 6

A programmer who wrote a function below thinks that the function tests if a given positive integer n>=2 is a prime. However, there is a logical mistake.

1. Where is the mistake?
2. For what number does the function return a wrong answer? Can you think of the smallest such number?
3. Fix the mistake.

```
def is_prime(n):
    '''(number)->bool
    Returns True if a given positive number n>=2  is prime and false otherwise
    Precondition n>=2
    '''
    for divisor in range(2, n//2+1):
        if n % divisor == 0:
            return False
        else:
            return True
    return True
```

Programming Exercise 3

Write a function first_neg that takes a (possibly empty) list of numbers as input parameter, finds the first occurrence of a negative number, and returns the index (i.e. the position in the list) of that number. If the list contains no negative numbers or it is empty, the program should return None. Use while loop (and not for loop) and your while loop should stop looping once the first negative number is found.

Test your function with at least the examples below

```
>>> first_neg([2, 3, -1, 4, -2])
2
>>> first_neg([2, 3, 1, 4, 2])
>>> first_neg([])
```

Programming Exercise 4

Write a function sum_5_consecutive that takes a list of numbers as input and returns True if there are 5 consecutive numbers in the list that sum to zero. Otherwise it returns False. The function should also return False if the list has less than 5 elements

Solve this in two ways:

1.for loop (over indices of the list)

2.while loop (over indices of the list)

In both cases you need to think about "stopping condition" in order to avoid "`IndexError: list index out of range`"

3. Test your function with at least the examples below

```
>>> sum_5_consecutive([2, 3, -3, 2, 4,-6])
True
>>> sum_5_consecutive ([-10, 1, 1, 4, 2, 10, 13])
False
>>> sum_5_consecutive([2, 1, -3, -3, -3, 2, 7, 4, -6])
True
>>> sum_5_consecutive ([])
False
>>> sum_5_consecutive ([1, -1,0])
False
```

Programming Exercise 5 : Different ways to create useful lists

Recall, for example, that
a=[2] creates a list (refer to by variable a) with one element, a number 2 in this case.
b=[None] creates a list with one element. That element is an object of type None. That is sometimes useful when we are not ready yet to assign a value to an element of a list.
c=[] creates an empty list, i.e. a list of length zero

Recall further that multiplying a list by an integer n, creates a new list that repeats the given list n times. Or that applying '+' operator to two lists creates a new lists that concatenates the given two lists. For example:
[1,2]+[10,20,0] creates a list [1,2,10,20,0]
[7,2]*3 creates a list [7,2,7,2,7,2]
Finally, recall the slicing. For example, if a=[2,3,4,1], a[:] returns a new list that is a copy of list a.

Open the file called creating_various_lists.py. The first line is given to you. It asks the user to enter a positive even integer n. For each green programing exercise below, try to find at least two solutions (e.g. one by using a loop with accumulator pattern and another by using int*list).

1.Create a list a (i.e a list referred to by variable a) of length n filled with zeros
2.Create a list b of length n filled with random integers between 1 and 100
3.Create a variable c that is an alias of b
4.Set first half of the elements of c to zero, and then print both b and c
5.Copy list b into a list d
6.Create a new list e that has every 2nd element of b

Programming Exercise 6: Fibonacci Numbers

Write a function called fib(n) that takes as input an integer n (greater than 1) and creates a list containing n values such that

a[0] = 1

a[1] = 1

a[i] = a[i-1] + a[i-2] for all i in the range 1 < i < n

and it prints that list once it creates it

```
>>> fib(7)
[1   1   2   3   5   8   13]
```

Once you are done, trace your program on paper and then Python visualizer to see what the values in the list will be for n=5.

Programming Exercise 7:

Write a function inner_product that that takes as input two lists of integers (of same length) and then computes and returns the *inner product* of the two lists of integers. The inner product of two lists $x = [x_1, x_2, \ldots, x_n]$ and $y = [y_1, y_2, \ldots, y_n]$ is the value

$x_1 y_1 + x_2 y_2 + \ldots + x_n y_n$.

```
>>> inner_product([2,3,4], [1,0,2])
10
```

More programming exercises (outside of lab):

Do everything in the first 13 exercises here

https://runestone.academy/ns/books/published/thinkcspy/Lists/Exercises.html

****

and if you need more exercises ….

go to the web site below, for some more practice programming questions but be aware that the web site is in Python 2, thus if you do these exercises do them on your own computer AND NOT in the little browser window of the web site.

https://leetcode.com/problemset/all/

http://codingbat.com/python