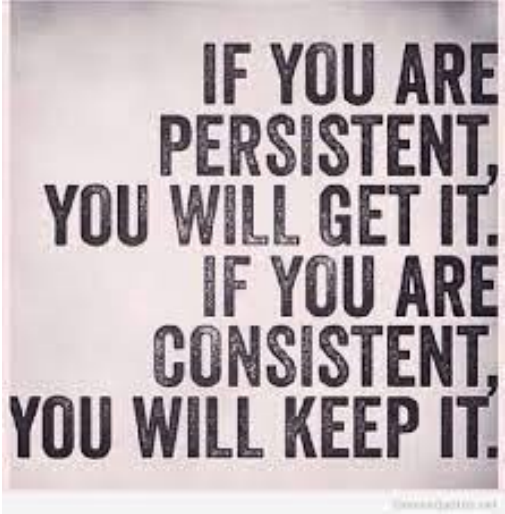
CS 152: More Functions

CS 152: Python for STEM

Colorado State University
Computer Science Department
Slides Originally Created by Albert Lionelle and Updated by Marcia Moraes

Colorado State University

Weekly Announcements!



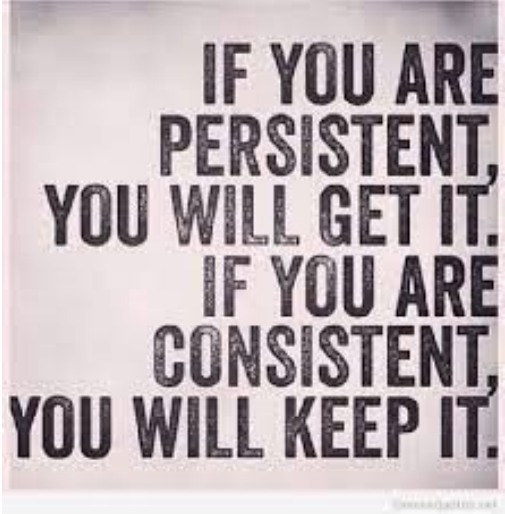
TODO Reminders:

- Reading 12 (zybooks) – you should have already done that ☺
- Lab 08
- Reading 13 (zybooks)
- Coding Exam 2
- Exam2

Recall Activity

What is the difference between local variables and global variables? Explain with your own words.

Have this written in a paper, you will turn in at the end of the class.



Scope of Variables

```
def numberGenerator(start, end, step):
    lst = []
    while start < end:
        lst.append(start)
        start += step
    return lst


start = int(input("Enter start value:"))
end = int(input("Enter end value:"))
step = int(input("Enter step value:"))
print(numberGenerator(start, end, step))
```

Local variables – declared inside of a function – exist while the function is being executed

Global variables – declared outside of a function

Namespaces

- A namespace maps names to objects

```
print('Initial global namespace: ')
print(globals())

my_var = "This is a variable"
print('\nCreated new variable')
print(globals())

def my_func():
    pass

print('\nCreated new function')
print(globals())
```

```
Initial global namespace:
{}

Created new variable
{'my_var': 'This is a variable'}

Created new function
{'my_func': <function my_func at 0x2349d4>, 'my_var': 'This is a variable'}
```

Scope Resolution

- There are at least three nested scopes that are active at any point in a program's execution:
  - Built-in scope: contains all of the built-in names of Python, such as int(), str(), list(), range(), etc.
  - Global scope: contains all globally defined names outside of any functions.
  - Local scope: Usually refers to scope within the currently executing function, but is the same as global scope if no function is executing.

- When a name is referenced in code, the local scope's namespace is the first checked, followed by the global scope, and finally the built-in scope.

Multiple Function Outputs

- Occasionally a function should produce multiple output values.
- However, function return statements are limited to returning only one value.
- A workaround is to package the multiple outputs into a single container, commonly a tuple, and to then return that container.

Multiple Function Outputs

```
def numberGeneratorAndSum(start, end, step):
    lst = []
    sumLst = 0
    while start < end:
        lst.append(start)
        start += step
        sumLst += start
    return lst,sumLst

start = int(input("Enter start value:"))
end = int(input("Enter end value:"))
step = int(input("Enter step value:"))
lst, sumLst = numberGeneratorAndSum(start, end, step)
print(lst)
print(sumLst)
```

Returning a tuple, could be also represented as (lst,sumLst)

Coding Activity

- With a Peer:
  - Write a Python function that receives two lists as a parameter and generates and returns other two lists. Remember that you need to return a tuple containing the lists.
  - One of the lists to be returned contains the sum of the elements in the same index for the lists that were passed as a parameter
  - The other list to be returned contains the multiplication of the elements in the same index for the lists that were passed as a parameter
  - You need to check the size of both lists passed as a parameter and you will consider the size of the shorter list to build the two new lists
    - Think about why do you need to do that…

Coding Activity

- With a Peer:
  - Write a Python function that receives a list as a parameter and calculates and returns the number of positive, negative, and zeros in that list. Remember that in order to return more that one value, you need to return a tuple.
  - Write a Python function that receives a list as a parameter and generates and return two lists, one containing the even values and the other containing the odd values from the original list. Remember that in order to return more that one value, you need to return a tuple.