# CMSC 201 Spring 2016
## Lab 05 – While Loops

**Assignment:** Lab 05 – While Loops
**Due Date:** During discussion, March 7th through March 10th
**Value:** 10 points

### Part 1: While Loops

A **`while`** loop statement in the Python programming language repeatedly executes a target statement as long as a given Boolean condition is **`True`**.

The syntax of a **`while`** loop in the Python programming language is:

```
while CONDITION:
    STATEMENTS(S)
```

Here, **`STATEMENT(S)`** may be a single statement or a *block* of statements. The condition can be any expression, as long as it evaluates to either **`True`** or **`False`**. (Remember, any non-zero value is seen as "**`True`**" by Python.) The **`while`** loop continues to run as long as (while) the condition is still **`True`**.

As soon as the condition evaluates to **`False`**, program control passes to the line of code immediately following the statements inside **`while`** loop. This is the first line of code after the **`while`** loop and its statements that it indented to the same depth as the "**`while CONDITION:`**" line of code.

Remember that in Python, all the statements indented by the same number of character spaces after a programming construct are considered to be part of a single *block* of code. Python uses indentation as its method of grouping statements.

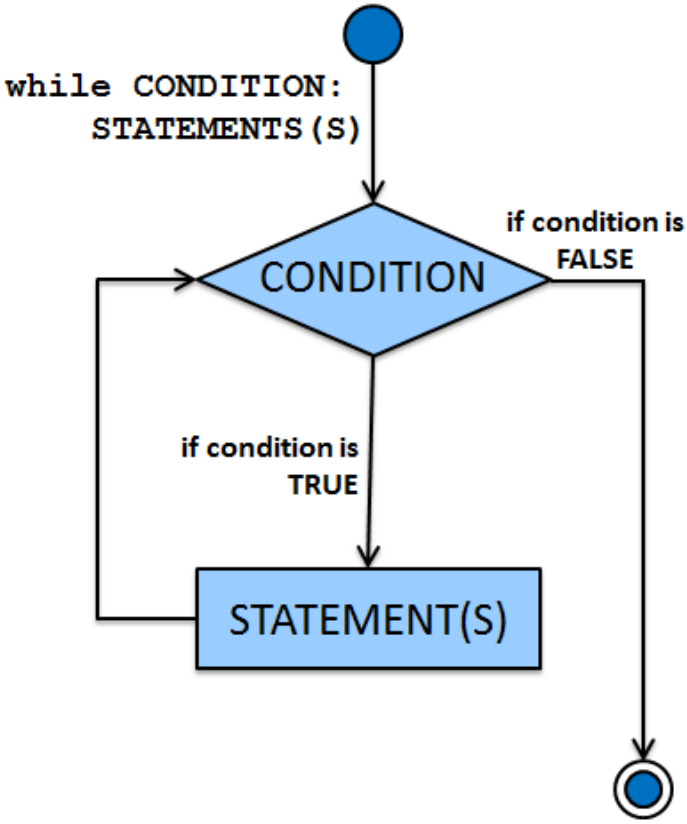


**Figure1. A `while` loop in Python**

It is also possible that a **`while`** loop might not ever run its conditional code (the "**`STATEMENT(S)`**" inside the **`while`** loop). If the condition is tested and the result is **`False`**, the loop body (the statements) will be skipped and the first line of code after the while loop will be executed.

```
count = 0
while (count < 5):
    print ('The count is:', count)
    count = count + 1
print ("Goodbye!")
```

When the above code is executed, it produces the following result :

```
The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
Goodbye!
```

## Part 2: Interactive (Sentinel) Loops

Another way to use a **`while`** loop is as an *interactive* or *sentinel* loop. An interactive loop continues to process data until reaching a special value that signals the end. The special value is called the *sentinel*.
You can choose any value for the sentinel. The only requirement is that it must be distinguishable from actual data values. It is also important that the sentinel is not processed as regular data (e.g., stored at the end of a user-created list, or included in the final calculation).

Here is the pseudocode for an interactive loop in Python:

```
Get the first data item from the user
While data item is not the sentinel:
    Process the data item
    Get the next data item from the user
```

One of the scenarios in which we can implement this type of loop is a version of our grocery list program that allows us to enter as many items as we like. Although it is similar to previous versions, the interactive (sentinel) while loop of the grocery list program allows us to enter as many items as we like until the sentinel value of **`"exit"`** is entered.

```
def main():
    grocery_list = []   # initialize the list to be empty
    # get the initial user value
    userVal = input("Enter an item, or 'exit' to end: ")

    # run the while loop until the user enters "exit"
    while userVal != "exit":
        grocery_list.append(userVal)
        # get another value from the user
        userVal = input("Enter an item, or 'exit' to end: ")

    # once the user is done with the list, print it out
    for g in grocery_list:
        print("Remember to buy", g)

main()
```

When the above code is executed, it produces the following result (with user input in blue):

```
Enter an item, or 'exit' to end: candy
Enter an item, or 'exit' to end: cookies
Enter an item, or 'exit' to end: gummy bears
Enter an item, or 'exit' to end: exit
Remember to buy candy
Remember to buy cookies
Remember to buy gummy bears
```

## Part 3A: Writing Your Program

After logging into GL, navigate to the **Labs** folder inside your **201** folder. Create a folder there called **lab5**, and go inside the newly created **lab5** directory.

```
linux2[1]% cd 201/Labs
linux2[2]% pwd
/afs/umbc.edu/users/k/k/k38/home/201/Labs
linux2[3]% mkdir lab5
linux2[4]% cd lab5
linux2[5]% pwd
/afs/umbc.edu/users/k/k/k38/home/201/Labs/lab5
linux2[6]%
```

(You will only be writing one python file for this assignment. )

To open the file for editing, type

**emacs shopping.py**

and hit enter.

The first thing you should do in your new file is create and fill out the comment header block at the top of your file. Here is a template:

```
# File:          shopping.py
# Author:        YOUR NAME
# Date:          TODAY'S DATE
# Section:       YOUR DISCUSSION SECTION NUMBER
# E-mail:        USERNAME@umbc.edu
# Description:   YOUR DESCRIPTION GOES HERE AND HERE
#                YOUR DESCRIPTION CONTINUED SOME MORE
```

Now you can start writing your code for the lab, following the instructions in Part 3B.

## Part 3B: Shopping List

To practice using **`while`** loops and lists, you will be creating a program that fills a shopping list with items from the user and then empties that list while asking for, and totaling, the costs for each item.

For the first part of your program, you should use a **`while`** loop and **`input`** statements to fill a list with items for a shopping trip. You should continue to prompt for an item until the user enters "**`done`**" for their choice. When finished, the list should be printed to the user.

To add items to your shopping list, you should use the **`append()`** function, as discussed in class. You can use **`print(shoppingList)`** to print the entire list, including the brackets and quotation marks.

For the second part of your program, you should ask the user for the price of each item, cross that item offthe list, and print out the total cost once the price of all of the items have been entered. You should also print out the list at the "end" of the shopping trip (it should be empty).

You may assume there is no sales tax when presenting the total. To cross items off your shopping list, you should use the **`remove()`** function, as discussed in class.

A sample run, with user input in blue, can be seen on the next page.

There are also hints on the next page, if you need them.

```
bash-4.1$ python shopping.py
Add item to list ('done' when finished): pocky
Add item to list ('done' when finished): ice cream
Add item to list ('done' when finished): orange juice
Add item to list ('done' when finished): cereal
Add item to list ('done' when finished): ramen
Add item to list ('done' when finished): done
Final shopping list:  ['pocky', 'ice cream', 'orange
juice', 'cereal', 'ramen']

How much did pocky cost? 1.99
How much did ice cream cost? 5.79
How much did orange juice cost? 4.66
How much did cereal cost? 3.50
How much did ramen cost? .30

Your shopping trip cost $16.240000000000002
Shopping list at end of trip:  []
```

**Try to solve Part 3B on your own before you turn to these hints!**

Are you stuck on how to get started?
*Start by creating the interactive (sentinel) loop that gets the user to enter an item. Get the loop to work (stopping when the user enters "done") before you worry about anything else.*
***If you accidentally made an infinite loop, use CTRL+C to stop it running!***

Having trouble with keeping track of the prices and removing items?
*You should use a variable to keep track of a "running total" – the variable should ben initialized before the loop begins. After getting the price of an item, you should remove it from the list. The loop should end when the list is empty.*

Not sure how to remove items from the list?
*You will need to use a while loop, not a for loop. It should be used in conjunction with the remove() function. If you're still stuck, make sure the conditional your while loop uses will stop at the correct point.*

## Part 4: Completing Your Lab

To test your program, first enable Python 3, then run **`shopping.py`**. Try a few different inputs to see how well your program works.

(For example, here is a sample run with an "empty" shopping list.)

```
bash-4.1$ python shopping.py
Add item to list ('done' when finished): done
Final shopping list:  []


Your shopping trip cost $0.0
Shopping list at end of trip:  []
```

Since this is an in-person lab, you do not need to use the **`submit`** command to complete your lab. Instead, raise your hand to let your TA know that you are finished.

They will come over and check your work – they may ask you to run your program for them, and they may also want to see your code. Once they've checked your work, they'll give you a score for the lab, and you are free to leave.

**IMPORTANT:** If you leave the lab without the TA checking your work, you will receive a **zero** for this week's lab. Make sure you have been given a grade before you leave!