Flow Control
(while Loop and break and continue Statement)

while Loop

- Do something WHILE a condition is **True**.
- Loop will STOP when condition is **False**.
- **break** statement is executed to exit the while loop.
- when **continue** statement is executed, control returns to beginning of loop.

REVIEW: Find the SUM of the numbers from 1 to 10.

```
n = 10
sum = 0
i = 1
while i <= n:
    sum = sum + i
    i = i + 1
print("The sum is ", sum)
```

sum = sum + i
Variable Sum to hold the numbers that are being counted

i = i + 1
Variable i to increment from 1 to 2 to 3 to 4 up to 10.

print("The sum is ", sum)
Will print the final sum outside of the while loop

Find the SUM of the numbers from 1 to 100000.

```
theSum = 0
count = 1
while count <= 100000:
    theSum = theSum + 1
    count = count + 1
print(“The sum is “, theSum)
```

theSum = theSum + 1
Variable theSum to hold the numbers that are being counted

count = count + 1
Variable count to increment from 1 to 2 to 3 to 4 up to 100000.

print(“The sum is “, theSum)
Will print the final sum outside of the while loop

TASK:

Find the SUM of numbers entered from a user, or just press enter to stop. Loop will stop when user presses <enter> (empty string “”)

REVIEW: Pseudocode Algorithm

```
set the sum to 0.0
input a string
while the string is not the empty string
        convert the string to a float
        add the float to the sum
        input a string
print the sum
```

```
theSum = 0.0
data = input(“Enter a number or just press enter to quit: “)
while data != “”:                              #empty string “”
        number = float(data)
        theSum = theSum + number
        data = input(“Enter a number or just press enter to quit: “)

#The next statement is outside the loop
print(“The sum is “, theSum)
```

Enter Code: today’s_date.py

while Loop using a break Statement

TASK:

Find the SUM of numbers entered from a user, or just press enter to stop.

```
theSum = 0.0
while True:
        data = input(“Enter a number or just press enter to quit: “)
        if data == “”:                                   #empty string “”
                break
        number = float(data)
        theSum = theSum + number

#The next statement is outside the loop
print(“The sum is “, theSum)

#The break Statement will cause an exit from the while loop.
```

TASK:

Enter a numeric grade from 0 – 100. Make sure to handle any invalid numeric grades.

Enter Code: today’s_date.py

if number >= 0 and number <= 100

is True, **break** will exit out

of the while loop

```
while True:
        number = int(input(“Enter a numeric grade from 0 – 100 ”))
        if number >= 0 and number <=100:
                break
        else:
                print(“Error: grade must be between 0 – 100 “)

print(“The grade is “, number)
```

#If a user enters a number from 0 to 100, the if condition will be True and the break Statement will cause an exit from the while loop.

#If a user doesn’t enter a number from 0 - 100, the else block will execute and the Error Message will print, and the while loop continues prompting the user for a numeric grade again.

```
var = 10

while var > 0:
        print(“Current Value: “, var)
        var = var – 1
        if var == 5:
                break                           #if condition var == 5 is True, break will exit out
                                                of the while loop

print(“Good bye!”)
```

Enter Code: today’s_date.py

while Loop using a continue Statement

#The **continue** statement returns the control to the beginning of the while loop.

#The **continue** statement rejects all the remaining statements in the loop.

```
var = 11
while var > 0:
        var = var – 1
        if var == 5:
                continue
        print(“Current Value: “, var)

print(“Good bye!”)
```

#if condition var == 5 is True, **continue** will return back to the beginning of the loop.