Lecture 13
Loop Examples

Objectives

- Loop revision
- Break statement
- Continue statement
- Loop examples

for Loop: Revision

```
for i in range(10):
   # do something
#------------------------------------------------------
myList = [2,3,4,9,10]
for x in myList:
   # iterates through the list elements
   # do something that involves the list elements
#------------------------------------------------------
myString = “hello there, hello world!”
for ch in myString:
   # iterates through the string characters
#------------------------------------------------------
infile = open(someFile, "r")
for line in infile:
   # iterate through the lines of the file
infile.close()
```

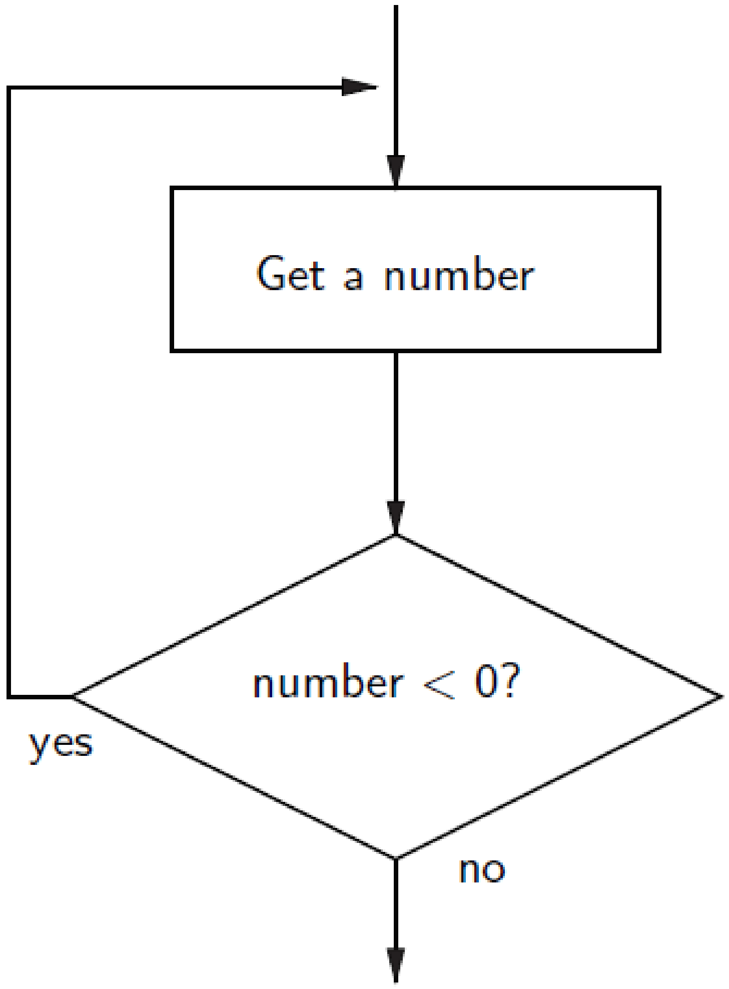
while loop: revision

```
while <condition>:
   # do something
#-------------------------------------------------
# program to list first 10 numbers
# valid but poor use of while

i = 0
while i < 10:
   print(i)
   i += 1
#-------------------------------------------------
# program to guess a secret number
n = 7 # secret number
guess = 1
while guess != n:
   guess = int(input("Please guess a number between 0 and 10"))
print("You guessed correctly")
```

Post-Test Loop

```
repeat
    get a number from the user
until number is >= 0
```



Nested Loops

- Designing nested loops –
  - *Design the outer loop without worrying about what goes inside*
  - *Design what goes inside, ignoring the outer loop.*
  - *Put the pieces together, preserving the nesting.*

Loop and a Half

- Stylistically, some programmers prefer the following approach:

```
while True:
    number = float(input("Enter a positive number: "))
    if number >= 0: break # if valid number exit loop
    print("The number you entered was not positive")
```

- Here the loop exit is in the middle of the loop body. This is what we mean by a *loop and a half*.

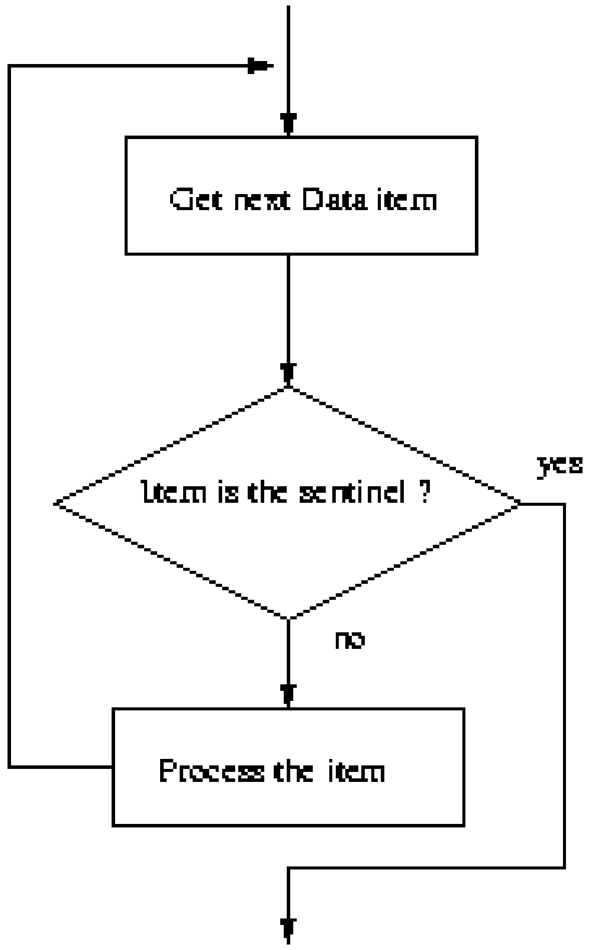
Loop and a Half

- The loop and a half is an elegant way to avoid the priming read in a sentinel loop.

```
while True:
    # get next data item
    # if the item is the sentinel: break
    # process the item
```

- This method is faithful to the idea of the sentinel loop, the sentinel value is not processed!

Loop and a Half



Loop and a Half

- To use or not use `break`. That is the question!
- The use of `break` is mostly a matter of style and taste.
- Avoid using `break` often within loops, because the logic of a loop is hard to follow when there are multiple exits.

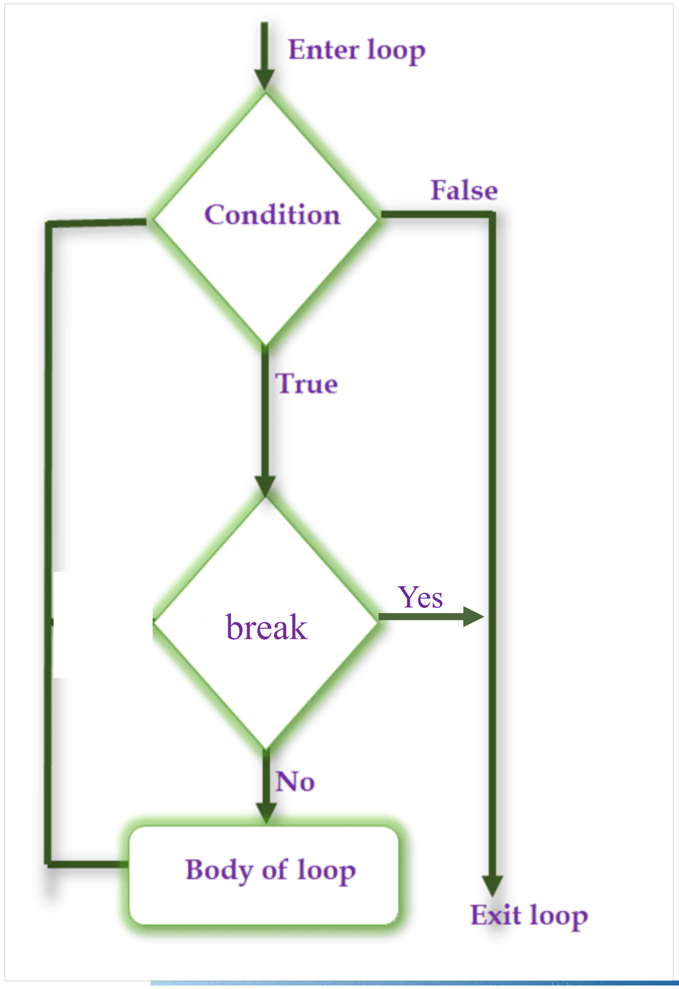
continue statement

- Continue statement returns the control to the beginning of the loop

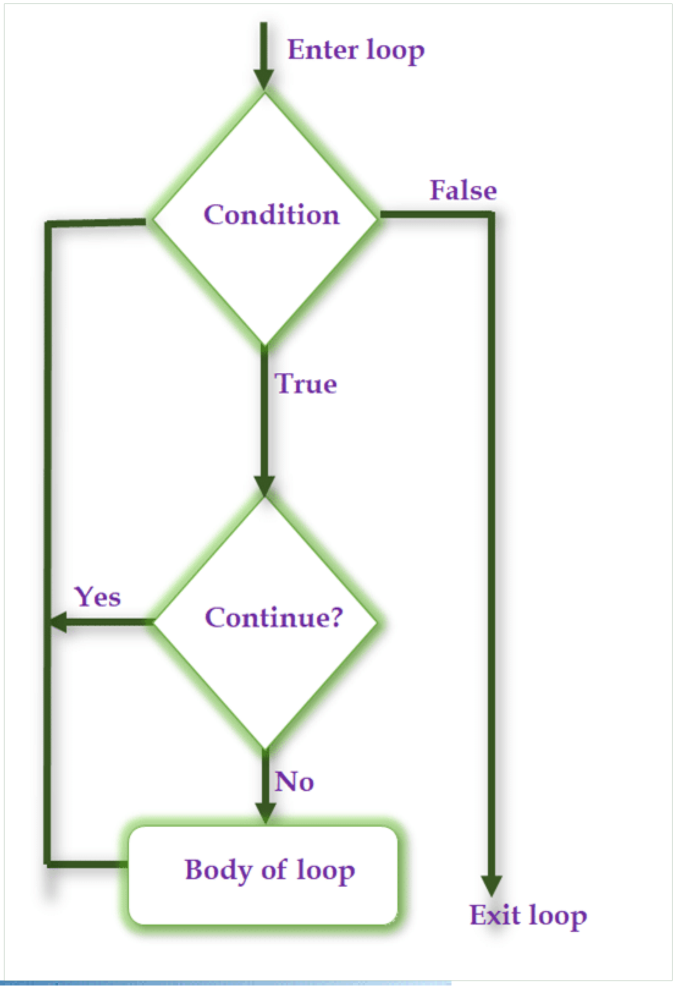
```
# print only even numbers up to 10
for i in range(11):
    if i % 2 == 1:  # % is "modulus" operator
        continue
    print(i)
```

break and continue comparison

break statement



continue statement



Loop Example: Prime number

- Find whether a number is prime or not
- Find list of prime number up to N
- Find N prime numbers
- Find N prime numbers using break

Finding whether a number is prime or not ?

```
def primestatus(N):
    if N < 2:
        return False
    elif N < 4:
        return True
    else:
        for i in range(2,N//2+1):
            if N % i == 0:
                return False
        return True
```

Find list of prime number up to N

```
def primelist(N):
    if N < 2:
        return []
    elif N == 2:
        return [2]
    else:
        plist = [2,3]
        status = True
        for num in range(4,N+1):
            for i in range(2,num//2 + 1):
                if num % i == 0:
                    status = False
            if status:
                plist.append(num)
            status = True
        return plist
```

Find N prime numbers

```
# Find the first N prime numbers
# Author: Michael J Wise
def primes(N) :
  primelist = [2,3]
  for pno in range(2,N) :
    i = primelist[-1] + 2 # start search for next one where
    primefound = False    # where last one left off
    while not primefound : # Test successive odd numbers
      factorfound = False
      for divisor in primelist :  #Only use previous primes
        if i % divisor == 0 :
          factorfound = True
      if factorfound :  # not prime
         i += 2
      else :
         primelist.append(i)
         primefound = True
  return(primelist)
```

Finding N primes numbers – with `break`

```
# Find the first N prime numbers (further optimised)
import math
def primes(N) :
  primelist = [2,3]
  for pno in range(2,N) :
    i = primelist[-1] + 2 # start search for next one where left off
    while True :
      factorfound = False
      if N > 100 :   # time for sqrt not worth it for N<=100
        stopat = int(math.sqrt(i))
      for divisor in primelist : # Only test previous primes
        if N > 100 and divisor > stopat :
          break  # From divisor search loop
        if i % divisor == 0 :
          factorfound = True
          break
      if factorfound :  # not prime, keep searching
        i += 2
      else :
        primelist.append(i)
        break  # Got a prime, break from this prime search
  return(primelist)
```

Summary

- break statement
- continue statement
- Example: prime numbers