Chapter 8

Loops structures and booleans

We have used for loop. The for loop iterates a well known number of times.

```
def Main()
    list = [3, 2, 5, 1]
    sum = 0
    for i in range (len(list))
        sum = sum + list[i]
    print("sum = ", sum).

Main()
```

Runs 4 times = len(list)

Sometimes we don't know in advance the number of times the loop is going to run.

Indefinite Loop

The while loop will iterate as long as a condition is true.

```
while <condition>:
    <body>
```

As long as the <condition> is true <body> will run.

You can convert a "for" loop to an while loop:

```
def main():
    list=[3,2,5,1]
    sum=0
    i=0
    while i < len(list):
        sum = sum + list[i]
        i = i+1
```

```
print("sum=", sum
Main()
```

The "for" loop is prefered to the "while" loop since it is easier to use. However there are situations when we don't know how many times the loop will run. In this case we use the while loop.

Example

- Write a program that computes the average.
- It will ask for a list of numbers
- computes average and prints it
- It will ask if you want to continue.

```
def average(l):
    sum=0
    for x in l:
        sum = sum + x
    avg= sum/len(l)
    return avg
```

equivalent

```
for i in range(len(l)):
    sum= sum + l[i]
```

```
def Main():
    yesno= "yes"
    while yesno == "yes":
        nums = eval( input("Type numbers x0,x1...:"))
        avg = average(nums)
        print ("average= ", avg)
        yesno = input("Do you want to continue yes/no")
    print("Thank for using average calculator")
```

"break" statement

- The break statement allows you to exit the loop. ~~anywh~~ It can be called anywhere inside the body of either a "for" loop or a "while" loop.
- In this way it does not need to finish the current iteration.

Example

Finding prime numbers

A prime number is a number that can be divided only by 1 and itself.

```
#Function that returns True if the
#number is prime or False, if it is not

def isPrime(x):
    prime=True
    for i in range(2,x):
        if x%i==0:
            prime=False
            break
    return prime
```

2, 3, 4, 5, 6, 7, 8, 9, 10

2, 3, 4, 5, 6, 7, 8, 9

exits the for loop it will not continue testing other i's. set prime to False

if it reaches x-1, then prime will keep the True value so it is a prime

```
def Main()
    print("Is 5 a prime number?", isPrime(5))
    print("Is 10 a prime number?", isPrime(10))
Main()
```

True

False.

---

① We can modify the Main() program to print all prime numbers less than some number N.

```
def Main():
    print("Print prime numbers less than N")
    N = eval(input("N ?"))
    for i in range(1, N):
        if isPrime(i):
            printf(i, "is a prime number")
```

N → 1000

i → 1, 2 1000

② We can modify again the program to print the first (M) prime numbers.

```
def Main():
    nprimes = 0      # prime numbers found
    i = 1            # Number tested so far to
                     # be a prime number.

    print("Find the first M prime numbers")
    M = eval(input("M ?"))

    while nprimes < M:
        if isPrime(i):
            print(i, "is a prime number")
            nprimes = nprimes + 1
        i = i + 1
```

Infinite loops


```
while True:
    <body>
```

You can use "break" to exit from the loop at any time.

Example:
Run the M primes number program and ask if you want to continue or not.

```
def Main():
    while True:
        { code to print first M prime numbers
          written before }
        yesno = input("Do you want to continue yes/no")
        if yesno == "no":
            break
```