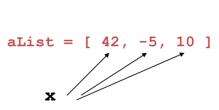
While Loops in Python

A BIT UNFINISHED BUSINESS WITH FOR LOOPS …

Two kinds of `for` loops

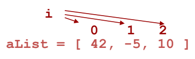
Element-based Loops

```
sum = 0

for x in aList:
    sum += x
```



Index-based Loops

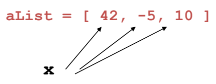
```
sum = 0

for i in                :
    sum +=
```



Two kinds of `for` loops

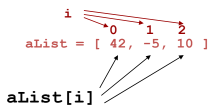
Element-based Loops

```
sum = 0

for x in aList:
    sum += x
```

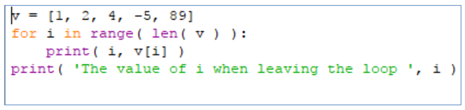


Index-based Loops

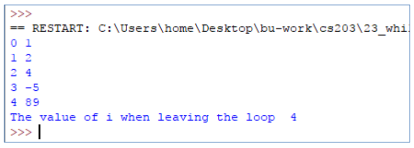
```
sum = 0

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



Run loopindex.py



```
v = [1, 2, 4, -5, 89]
for i in range( len( v ) ):
    print( i, v[i] )
print( 'The value of i when leaving the loop ', i )
```



```
>>>
== RESTART: C:\Users\home\Desktop\bu-work\cs203\23_whi
0 1
1 2
2 4
3 -5
4 89
The value of i when leaving the loop  4
>>> |
```

Python ‘for’ loop works well for a collection of known items. That is, we know the count, or we know the list of items.

But what if we don’t know the count, or the list, rather we only know the condition to repeat (or not to repeat) the steps? For example, we want to repeat adding to a sum as long as the user input is a positive number.

Here is a problem to solve:
The user types in a sequence of positive integers. The program is to add up the integers. The user signals the end of the input by entering a non-positive value which should not be part of the sum.

```
sum = 0
v = int( input( ‘Enter a positive integer ( <= 0 to stop) : ‘))
while ( v > 0):
  sum = sum + v
  v = int( input( ‘Enter a positive integer ( <= 0 to stop) : ‘))
print( ‘Summation is : ‘, sum )
```

while <condition>:
<body>

`while` loops: *indefinite, conditional* iteration

```
x = 10
if x != 0:
   print(x)
   x = x – 1
print(x)
```

What are printed on the screen?

```
10
9
```

`while` loops: *indefinite, conditional* iteration

while <condition>:
<body>

```
x = 10
while x != 0:
   print(x)
   x = x – 1
print(x)
```

What is the last value printed on the screen?
a) 0
b) 1
c) 9
d) 10

Patterns of while loop

Prime the condition.

while <condition>: *Check the condition.*
<body>
<update condition> *Update the condition.*

```
x = 10
while x != 0:
   print(x)
   x = x – 1
print(x)
```

From logical point of view, a **while** loop must have three components.
1. Prime the condition; [before]
2. Check the condition; [in]
3. Update the condition. [in]

Missing any of the three likely makes the loop incorrect.

Extreme Looping

What does this code do?

```
print('It keeps on')

while True:
   print('going and')


print('Phew! I\'m done!')
```

Extreme Looping

Anatomy of a while loop:

```
print('It keeps on')

while True:
   print('going and')


print('Phew! I\'m done!')
```

the loop keeps on running as long as this test is **True**

“while” loop

This won't print until the while loop finishes - In this case, it ***never*** prints!

alternative tests?

Making our escape!

```
import random
escape = 0

while escape != 42:

   print('Help! Let me out!')
   escape = random.choice([41,42,43])


print('At last!')
```

Will the same thing appear every time this is run?

Count the iterations…

```
import random
escape = 0

while escape != 42:

   print('Help! Let me out!')
   escape = random.choice([41,42,43])


print('At last!')
```

What modifications do we need to make to count the number of iterations and report it?

how could we make it easier/harder to escape?

Count the iterations…

```
import random
escape = 0
count = 0

while escape != 42:
   count += 1
   print('Help! Let me out!')
   escape = random.choice([41,42,43])


print('At last!')
```

What modifications do we need to make to count the number of iterations and report it?

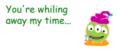
how could we make it easier/harder to escape?

Try out escape.py

while we're here…

What numbers will this loop print out?

```
x = 39
while x < 44:
    if x % 2 == 0:
        print(x)
    x += 1
```



Exercises

- Write a for loop to print the first n values of multiple of 3

- Write a while loop that takes a collection of positive integers and computes the sum. The loop stops when a zero or negative number is entered. Compute and print the sum and average of this collection of numbers.