`FOR` loops

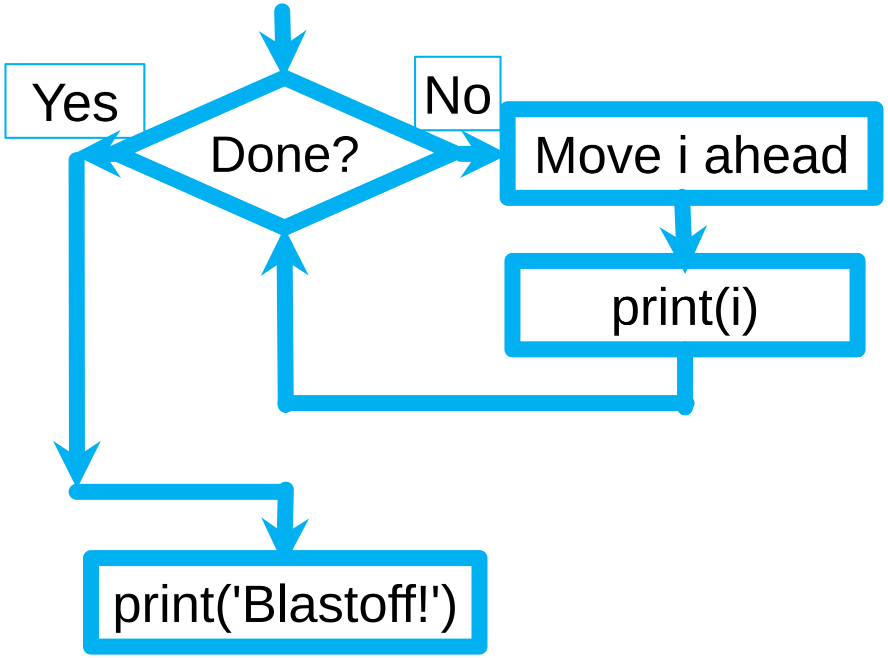
Lecture 04.02

By Marina Barsky

`for` loops:
definite, intentional
iteration

```
for x in [1,2,3]:
    print(x)
```

Program flow with **for**



```
for i in [5, 4, 3, 2, 1]:
    print(i)
print('Blastoff!')
```

```
5
4
3
2
1
Blastoff!
```

- Definite loops (**for** loops) have explicit iteration variables that change with each pass through a loop.
- These loops are called “definite loops” because they execute a **predefined number of times**

`for` with list of strings

```python
friends = ['Joseph', 'Glenn', 'Sally']
for friend in friends:
    print('Happy New Year:', friend)
print('Done!')
```

```
Happy New Year: Joseph
Happy New Year: Glenn
Happy New Year: Sally

Done!
```

Loops aren't just for lists...

```
for c in 'down with CS!':
    print(c)
```

```
d
o
w
n

w
...
```

We can loop over any *iterable* object

Iteration *variable*

1 **x** is assigned each value from this sequence in turn

```
for x in [5,4,3,2,1]:
    print('x is',x)

print('Blastoff!')
```

2 the BODY of the for loop runs with that value of **x**

3 LOOP back to step 1 for EACH value in the list

4 Code AFTER the loop: will run only after the loop is finished.

```
x is 5
x is 4
x is 3
x is 2
x is 1
Blastoff
```

for loop: syntax

for each element of the list – assign this element to variable x,
do something with this variable in the loop body

```
for x in [2,4,6,8]:
    print(x)


for c in [7]*6:
    print(c)


for n in                    :
    print(n)
```

How could we get this loop to run 42 times?

There are is ***range*** of answers to this one…

for loops: syntax

```
for x in [2,4,6,8]:
    print(x)

for c in [7]*6:
    print(c)

for n in range(42):
    print(n)
```

How could we get this loop to run 42 times?

Sum with for

```
def sum(a_list):
  answer = 0
  for x in a_list:
    answer = answer + x
  return answer
```

Factorial with for

```
def fac(n):
    answer = 1
    for x in range(1,n+1):
        answer = answer * x
    return answer
```

Iterating through sequences

- We have mostly been using the `in` keyword with `for` to access each element of the list

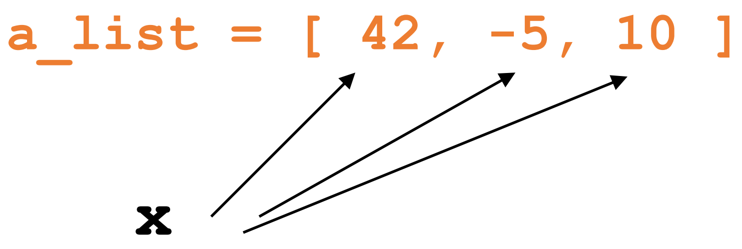
```
for x in [2,22,222,2222]
    print(x)
```

- There is another common approach…

Two kinds of **for** loops

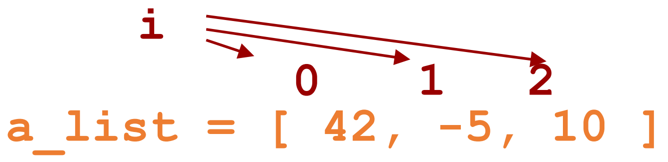
Element-based Loops

```
sum = 0

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



Index-based Loops

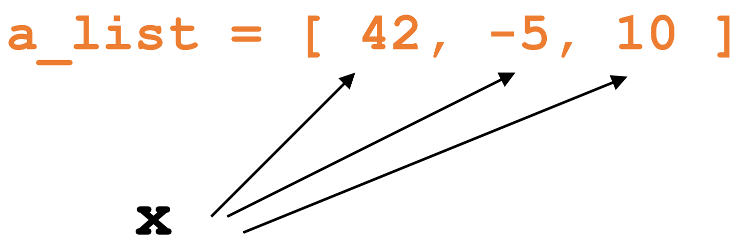
```
sum = 0

for i in                :
    sum += a_list[i]
```



Two kinds of `for` loops

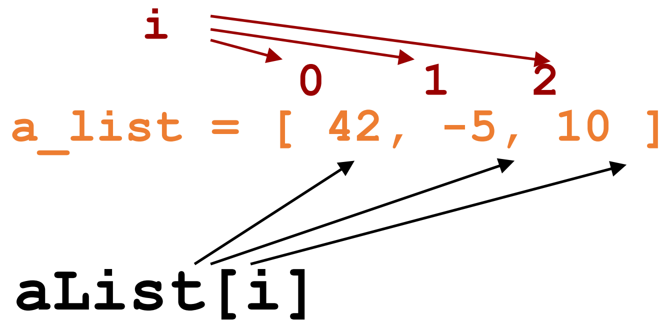
Element-based Loops

```
sum = 0

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



Index-based Loops

```
sum = 0

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



Summary

- We’ve learned how to perform a predefined number of iterations using **for** loop
- We can iterate over elements of a list or string, or we can iterate over indices
- To create a range of indices we use a new data type: range
- To produce a range we use function ***range()***

while vs. for

- You can simulate any for loop with a while loop

```
for i in range(n):
    <body of loop>
```

- is the same as

```
i=0
while i < n:
    <body of loop>
    i = i+1
```

what happens if this line is omitted?