Around and Around We Go
Loops and Iteration

Around and Around

- One of the most fundamental processes in a computer program is to repeat statements many, many (perhaps many, many, many) times.
- Computers never run out of patience
  - but loops are sometimes very slow compared to equivalent constructs, though this is mainly true in interpreted languages.

Loops

- Nearly every non-trivial program requires some form of looping or iteration. Examples:
  - Read all the lines in a file
  - Assign all values of an array
- In Python we have `for` and `while` loops.

Python for

- Python's for loop is in many ways quite different from a DO or the FOR in other languages (e.g. C/C++), but it is similar to Perl, bash, etc.

```
for <iterated item> in <iterator object>:
    block1
else:  # optional
    block2
```

- The else clause is executed when the loop completes the iterator.
- Colons are required as indicated. The blocks must be indented.

Range

- for loops are often used with lists, which we haven't yet covered.
- To loop over a sequence of numbers, use the `range` function.
  - Arguments must be integers!

`range(10)` : 0,1,2,3,4,5,6,7,8,9

`range(1,10)`: 1,2,3,4,5,6,7,8,9

`range(0,10,2)`: 0,2,4,6,8

`range(10,0,-2)`: 10,8,6,4,2 [NO ZERO!]

If the stride is present the lower-bound argument must be present. Otherwise the lower bound may be omitted, in which case it is zero. If the stride is omitted it is 1.

```
for i in range(10):
    print i
else:
    print “10”
```

Leaving Early

- What on Earth is the purpose of the else?
  - Might be to set final value
  - Mostly it's used with early exits

`break`: leave loop – any else clause will not be executed

`continue`: skip rest of loop and go to next iteration

```
j=2
for i in range(10):
    if ( i + j >= 10 ):
        m=12
        break
else:
    m=4
print m
```

WHILE Loops

- While loops use a conditional to determine when to exit
- Make sure your expression evaluation changes!!!

```
while <Boolean expression>:
    block
else: # optional
    block
```

Else clause is executed iff the conditional becomes False (normal termination).

- Equivalent to:

```
while True:
    if ( not <Boolean expression> ) exit
    statement
    statement
    ....
end do
```

NOTE: do while always tests at the *top* of the loop. The do … if/exit form can test anywhere, e.g. at the *bottom* to implement the repeat-until of some other languages.

- Python syntax

```
while <Boolean expression>:
    block
else: # optional
    block
```

Else clause is executed iff the conditional becomes False (normal termination).

Python

```
x=-20
y=-10
while (x<0 and y<0):
    x=10-y
    y=y+1
    z=0
else:
    z=1
```

Python Reading a File

```
for f in open("foo"):
        process input
```

- More examples and other ways to do it when we talk about file IO

Break/Continue

```
x=1.
while x>0.0:
    x=x+1.
    if x>=10000.0: break
    if x<100.0: continue
    x=x+20.0
```

Do Nothing (No-Op)

- Python

```
pass
```

Infinite loop:

```
while (True): pass
```

Python Repeat/Until

```
while True:
    statement
    statement
    statement
    if something: break
more statements
```