Loops

Iteration with `for` loops

---

CS112 Scientific Computation
Department of Computer Science
Wellesley College

---

Iteration

We often want to repeat an operation multiple times or step through a collection of values and perform the same computation for each value.

For example, drawing the olympic symbol, repeating the sphere in the perceptual illusion, cleaning up the ocean depth data, ...
Repetitive computations

Repetitive computations can be implemented with a for statement:

\[
\text{for variable name = vector of values} \\
\text{code statements to repeat} \\
\text{end}
\]

For example:

\[
\text{for count = 1:5} \\
\text{disp('Peter Piper picked a peck of pickled peppers');} \\
\text{end}
\]

Flow diagram

\[
\text{for count = 1:5} \\
\text{disp('Peter Piper picked a peck of pickled peppers');} \\
\text{end}
\]
Let’s turn `peterPiper` into a function

```matlab
function peterPiper
    % peterPiper
    % repeats a tongue twister 5 times
    for count = 1:5
        disp('Peter Piper picked a peck of pickled peppers');
    end
end
```

Modify `peterPiper` so that the number of repeats is an input:

```matlab
>> peterPiper(10)
Peter Piper picked a peck of pickled peppers
Peter Piper picked a peck of pickled peppers
Peter Piper picked a peck of pickled peppers
Peter Piper picked a peck of pickled peppers
Peter Piper picked a peck of pickled peppers
...
```

Further modifications to `peterPiper`

Modify the `peterPiper` function further so that the value of the count variable is incorporated into the printout:

```matlab
>> peterPiper(3)    % hummm?
Peter Piper picked 1 pecks of pickled peppers
Peter Piper picked 2 pecks of pickled peppers
Peter Piper picked 3 pecks of pickled peppers
```
Creating a bull’s eye display

function makeBullseye
% creates a display of blue concentric circles
% create 50 evenly spaced angles around a circle
angles = linspace(0, 2*pi, 50);
hold on
% plot 10 circles of increasing radius
for radius = 10:10:100
    plot(radius*cos(angles), radius*sin(angles));
end
axis equal
hold off

Variety is the spice of life

function makeBullseye2
% creates a display of multi-colored concentric circles
% create 50 evenly spaced angles around a circle
angles = linspace(0, 2*pi, 50);
% create a vector of different colors from a string
colors = 'bgrcmybgrc';
hold on
% plot 10 circles of increasing radius and changing color
index = 1;
for radius = 10:10:100
    plot(radius*cos(angles), radius*sin(angles), colors(index));
    index = index + 1;
end
axis equal
hold off
Better still?

How about a real Bull’s eye pattern with the colors filled in?

We can use the fill function instead of plot to create a Bull’s eye like this.

Bull’s eye!

```matlab
function makeBullseye3
    % creates a display of multi-colored concentric circles
    % create 50 evenly spaced angles around a circle
    angles = linspace(0, 2*pi, 50);
    % create a vector of different colors from a string
    colors = 'bgrcmybgrc';
    hold on
    % plot 10 circles of increasing radius and changing color
    for index = 10:-1:1
        fill(10*index*cos(angles), 10*index*sin(angles), colors(index));
    end
    axis equal
    hold off
end
```
Breaking out

There are times when we’d like to immediately exit a loop without stepping through all of the values of the control variable.

This can be done with a break statement.

Exercise: define a function collectGoldenRatios that continually prompts the user for hand and arm measurements, until the user enters a 0.

collectGoldenRatios

Modify collectGoldenRatios:
(1) prompt user up to 100 times for hand and forearm values and store ratios in a vector
(2) stop if the user enters a 0 for the hand length
(3) print message with number of measurements entered

function ratios = collectGoldenRatios
% ratios = collectGoldenRatios

disp('You will be prompted to enter hand and forearm lengths.

disp('When done, enter a 0 for the hand length.

hand = input('Enter a hand length ');
forearm = input('Enter a forearm length ');
ratios = forearm/hand;
Tip on debugging loops

% calculate 10! and print the result

factorial = 0;
for num = 10:1:1
    disp('inside loop');
    factorial = factorial * num;
    disp(['num: ' num2str(num) ' factorial : ' num2str(factorial)]
end
disp(['10! = ' num2str(factorial)]);