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
- cleaning up the depth data
- recognizing faces
Repetitive computations

Repetitive computations can be implemented with a `for` statement:

```matlab
for variable name = vector of values
code statements to repeat
end
```

For example:

```matlab
for count = 1:5
disp('Peter Piper picked a peck of pickled peppers');
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 the number of repeats is an input:

```matlab
>> peterPiper(4)
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 peterPiper function further so the value of the count variable is incorporated into the printout:

```matlab
>> peterPiper(3) hmm?
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

```matlab
function makeBullseye
% creates a display of 10 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
```

Boring…
Variety is the spice of life

function makeBullseye2
% creates a display of 10 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!

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

Breaking out

Sometimes we’d like to immediately exit a loop without stepping through all values of the control variable.

We can do this with a break statement:

num = 1;
for i = 1:100
    num = 2 * num;
    if (num > 100)
        break;
    else
        disp(['num' num2str(num)])
    end
end
collectGoldenRatios

Write a function named collectGoldenRatios:

(1) one input: maximum number of times to prompt the user for hand and forearm values

(2) “for loop” that prompts the user for hand and forearm values, for input number of times, and stores the ratios in a vector

(3) stop the loop if the user enters a 0 for the hand length

(4) print message at the end with number of measurements entered

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)]);