MATLAB
Patterns and Practices

Pete Benson

Department of Mathematics

530 Church Street

Ann Arbor, MI 48109-1043, USA

Chapter 1

Patterns

Each pattern is in a subsection, and the name of the pattern is the name of the subsection.

1.1 Repeating task n times

If you know you need to repeat something a specific number of times, use the for loop:

1.1.1 `for` loop

```
% for loop to sum numbers 1 to 5
sum_ = 0;
n = 5;
for iter = [1:n]
    sum_ = sum_ + iter;
end
fprintf('sum_ of 1 to %d is %d\n',n,sum_);
```

1.2 Repeating a task when you don't know in advance when you will be done

In short, if you need a loop and a `for` loop won't work, use `while`.

1.2.1 `while true` with `break`

A commonly used approach is to use an infinite loop structure (`while true`) with a `break` statement to get out of the infinite loop.

```
% add integers until the sum is greater than 100
sum_ = 0;
n = 0;
while true
    n = n + 1;
    sum_ = sum_ + n;
    if sum_ > 100
        break
    end
end
fprintf('sum of 1 to %d is %d\n',n,sum_);
```

1.2.2 `while <boolean expression>`

Alternatively, the `while` can execute conditionally, as long as the provided boolean expression evaluates to `true`.

```
% add integers until the sum is greater than 100
sum_ = 0;
n = 0;
while sum_ < 100
    n = n + 1;
    sum_ = sum_ + n;
end
fprintf('sum of 1 to %d is %d\n',n,sum_);
```

1.2.3 Validating user input

Getting correct input from the user is a perfect application of the `while true` with `break` pattern.

```
% get user input between 1 and 100
n = -1;
while true
    n = input('Enter a number from 1 to 100: ');
    if n >=1 && n <=100
        break
    end
end
fprintf('n = %d\n',n);
```

1.3 if, elseif, else,

1.3.1 if

If you want to do something only if some condition is met, and there is nothing special that needs to be done if the condition is not met, then you want the `if` statement.

```
% get age, and warn user if they need adult approval
age = input('Enter your age: ');
if age < 18
    fprintf('You will need adult approval.\n',n);
end
```

1.3.2 if-else

If you want to do a task if some condition is met, and you must do a different task if the condition is not met, use the `if-else` pattern. Note that `else` *does not* have a boolean expression attached to it.

```
% get age, and assign one of two categories
age = input('Enter your age: ');
age_category = '';
if age < 18
    age_category = 'minor';
else
    age_category = 'adult';
end
fprintf('For age = %d, category = %s\n',age, age_category);
```

1.3.3 `if-elseif`

If you must do one task from several possible tasks, use the `if-elseif`. Note that `elseif` *does* have a boolean expression attached to it.

```
% get age, and assign one of more than two categories
age = input('Enter your age: ');
age_category = '';
if age < 12
    age_category = 'child';
elseif age < 18
    age_category = 'youth';
else
    age_category = 'adult';
end
fprintf('For age = %d, category = %s\n',age, age_category);
```

Also, note that the last case (the default case) used `else` rather than `elseif`. Usually, this is what your last case will do.

Chapter 2

Practices

2.1 Habits

A list of good habits:

1. When you start writing a program, begin with comments describing (in sufficient detail) how your program will work.
2. Practice running your program in your head as you write. It may seem slow, but it is the fastest way to write working code.
3. Reformat your code frequently. This will help you detect missing end statements, and makes your code easier to understand.
4. When assigning value to a variable (e.g. `x = ...`), end the line with a semi-colon, unless you are debugging the value of `x`. This makes your output easier to read.
5. Typically, when using `fprintf`, insert \n. E.g. `fprintf('Hi \n')`.