Topics: Program design, user-defined function

Reading (ML): Ch 5 intro, Sec 5.1, 5.2

Programming Rules of Thumb

- *Learn program patterns* of general utility and *use relevant pattern* for the problem at hand.
- *Seek inspiration* by systematically working test data by hand. Be introspective; ask yourself: "what am I doing?"
- *Declare variables* for each piece of information you maintain when working problem by hand. *Write comments* that precisely describe the contents of each variable.
- *Decompose* problem into manageable tasks.
- *Remember* the problem's boundary conditions.
- *Validate* your program by tracing it on simple test data.

Program Trace

Trace the execution of the following program:

```
n=18;  x=3;  y=10;
while (n~=0)
    if (mod(n,2)==0)
        n = n/2;
    else
        n = n-1;
        x = 10*x+3;
        y = y*10;
    end
end
y = (y-1)/3;
```

n	18															
x		3														
y			10													

Time $\longrightarrow$

Example: Are they prime?

Write a program that saves in a vector all the prime numbers in the range of [2,**n**], (**n**¿1).

Script file **savePrime.m**:

```
% Save prime numbers from 2 thru n to vector prime

n = input('Enter number: ');
prime = 2;  % vector to store prime #s
i = 3;      % next number to be checked
while (i<=n)
  % call function to check number i, save to prime


  % get next number
    i = i+1;
end
prime
```

Function file **isPrime.m**:

```
% Determine if n is prime, n>=2
% out <-- n if n is prime
% out <-- [] if n is composite

divisor = 2;
while ( mod(n,divisor) ~= 0 )
  divisor = divisor + 1;
end
if ( divisor==n )


else


end
```

General Form of User-Defined Function

function *[outarg1, outarg2, ...]* = **fname**(*inarg1, inarg2, ...*)
`% H1 comment line`
`% Other comment lines`

executable code

User-Defined Function

- Can easily "reuse" code
- Functions can be independently tested
- Input and output arguments represent a "contract" between the developer and the user of the function
- Arguments are "passed by value"
- Variables in a function can be "seen" only inside the function

Be sure you understand the example on p. 164 in Chapman.