Day 22

- (Concept Question)
- Functions you know
- Functions made from a piece of a script
- Planning a new function
- Exercises

Functions you know

"function": special type of MATLAB script that takes an input from you and gives you an output


```
y = sin( x )
```

The output of the function is stored in the variable `y`

The name of the function is "`sin`"

The input to the function is `x`

Functions you know

"calling" the `sin` function (using it in the "main" script)

```
1 -    i = 1;
2
3 -    for theta = 0:15:90
4
5 -        sintheta(i) = sin(theta * pi/180);
6
7 -        i = i + 1;
8
9 -    end
```

`theta*pi/180` is the input to the function sin.m

The output from the function sin.m is stored in the variable `sintheta(i)`

Functions you know

Notice:

- Once function works, we don't need to see "inside" the script – we can use it over and over again
- We don't need to know what the variable names are in the function script – we pick the names of the inputs and outputs

Functions you know

Functions can have more than one output.

```
32      %
33      % Load the data file
34      %
35 -    load pressuredata
36
37 -    [Nrows,Ncols] = size(SCAN0);
38
39 -    for i = 1:Nrows
40
```

The matrix `SCAN0` is the input to the `size` function

The `size` function has two outputs—the number of rows and the number of columns

Functions you know

Functions can have inputs but no output variables

```
plot(x,y)
```

`plot` has two inputs but the only "output" is the plot itself

```
title('Rocket Acceleration')
```

`title` has one input and no output. It uses the input to make a plot title.

Functions you know

Functions can have no inputs and no output variables

```
clc
```

`clc` has no inputs but the only “output” is to clear the command window

Functions made from a piece of a script

You can create your own functions.

Often they are created from a piece of a program you want to use repeatedly

- Called several times from the same main script
- Called from several different main scripts

Functions made from a piece of a script

Example: This program calculates a factorial

```
1 -   clc
2 -   clear variables
3 -   n = input('Enter a number to calculate its factorial: ');
4
5 -   prod = 1;
6 -   for i = 1:n
7 -       prod = prod * i;
8 -   end
9
10 -  fprintf('The factorial of %2.0f is %5.0f \n', n, prod);
```

This piece of the code calculates the factorial. We might use that in other programs as well.

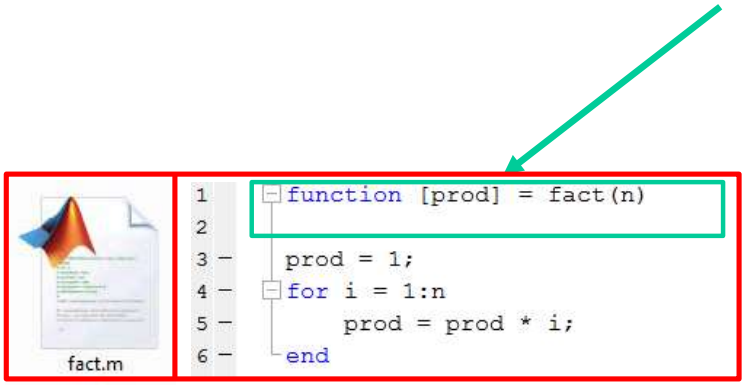
Functions made from a piece of a script

Take out the middle piece of code and place it in a new function called fact.m.

```
1 -   clc
2 -   clear variables
3 -   n = input('Enter a number to calculate its factorial: ');
4
5 -   prod = 1;
6 -   for i = 1:n
7 -       prod = prod * i;
8 -   end
9
10 -  fprintf('The factorial of %2.0f is %5.0f \n', n, prod);
```

Functions made from a piece of a script

Add a line in `fact.m` to tell MATLAB that this is a function.



This says the function name is `fact`, and it needs one input (`n`) and returns one output (`prod`).

Functions made from a piece of a script

SAVE THE SCRIPT `fact.m`

It does not save automatically.

It MUST be named `fact.m` or MATLAB won't know where to find the function `fact`.

Functions made from a piece of a script

Change the main script to call the function

```
1 -   clc
2 -   clear variables
3 -   n = input('Enter a number to calculate its factorial: ');
4
5
6 -   prod = fact(n);
7
8
9
10 -  fprintf('The factorial of %2.0f is %5.0f \n', n, prod);
```

`n` is the input to the function

The output is stored in `prod`

`fact` is the name of the function

Functions made from a piece of a script

Run the main script.

It does save automatically when you run it.

Never run the function by itself.

Functions made from a piece of a script

The variables in the main script and function are independent of each other—*you do not have to use the same variable names*.

```
1 -    clc
2 -    clear variables
3 -    x = input('Enter a number to calculate its factorial: ');
4
5
6 -    product = fact(x);
7
8
9 -    fprintf('The factorial of %2.0f is %5.0f \n',x,product);
```

The value of `x` is passed to the variable `n` in the function.

The value of `prod` is passed back to the variable `product` in the main script.


```
1      function [prod] = fact(n)
2
3 -      prod = 1;
4 -    for i = 1:n
5 -        prod = prod * i;
6 -    end
```


Functions made from a piece of a script

The original script and the new script with the function make exactly the same output.

Original program

```
1 -    clc
2 -    clear variables
3 -    n = input('Enter a number to calculate its factorial: ');
4
5 -    prod = 1;
6 -    for i = 1:n
7 -        prod = prod * i;
8 -    end
9
10 -   fprintf('The factorial of %2.0f is %5.0f \n', n, prod);
```

New program and function

```
1 -    clc
2 -    clear variables
3 -    x = input('Enter a number to calculate its factorial: ');
4
5
6 -    product = fact(x);
7
8
9 -    fprintf('The factorial of %2.0f is %5.0f \n',x,product);
```

Now we can use this function with other programs!


```
1      function [prod] = fact(n)
2
3 -      prod = 1;
4 -    for i = 1:n
5 -        prod = prod * i;
6 -    end
```

Planning a new function

Advantages of using functions in your scripts:

1. Your main script will be easier to understand
2. Big problems can be broken into smaller chunks.
3. Easier to debug smaller scripts than one large one
4. Avoid unnecessary duplication of segments of code—if you are cutting and pasting a portion of code several times, you probably should put it in a separate m-file as a function

Planning a new function

When planning a new function you must think of what inputs the function will need and what outputs it will produce.

```
[output1, output2, …] = functionname(input1, input2, …)
```

Planning a new function

```
14 -    clc
15 -    clear variables
16      % Define number of rectangles
17 -    N_rect  = 100;
18
19      % Define looping variable
20 -    t_start = 0.0;
21 -    t_end   = 5.0;
22 -    delta_t = (t_end - t_start) / N_rect;
23
24      % Open data file
25 -    file_number = fopen('Day6_Ex2.txt', 'w');
26
27      % Initialize velocity
28 -    velocity = 0.0;
29
30      % Computation loop
31 -    for num = 1:N_rect
32 -        t_instant = t_start + (num - 1) * delta_t;
33 -        acceleration = 0.2 * exp(2.1 * t_instant);
34 -        velocity = velocity + acceleration * delta_t;
35 -    end
36 -    fprintf(file_number, 'The velocity with %1.0f rectangles is %5.1f m/s \n',N_rect, velocity);
37
38      % Close data file
39 -    fclose(file_number);
```

As an example, let's consider the numerical integration program that you wrote for Day 6, Exercise 2—where we set how many rectangles you use for the integration approximation.

We might consider organizing this better by making the computation loop part of a function—then we could call the function with any number of rectangles we want.

Planning a new function

```
14 -    clc
15 -    clear variables
16      % Define number of rectangles
17 -    N_rect  = 100;
18
19      % Define looping variable
20 -    t_start = 0.0;
21 -    t_end   = 5.0;
22 -    delta_t = (t_end - t_start) / N_rect;
23
24      % Open data file
25 -    file_number = fopen('Day6_Ex2.txt', 'w');
26
27      % Initialize velocity
28 -    velocity = 0.0;
29
30      % Computation loop
31 -    for num = 1:N_rect
32 -        t_instant = t_start + (num - 1) * delta_t;
33 -        acceleration = 0.2 * exp(2.1 * t_instant);
34 -        velocity = velocity + acceleration * delta_t;
35 -    end
36 -    fprintf(file_number, 'The velocity with %1.0f rectangles is %5.1f m/s \n',N_rect, velocity);
37
38      % Close data file
39 -    fclose(file_number);
```

Let us put the numerical integration part of this code in a function.

IntegrateAcceleration.m

Planning a new function

The part of our code that integrates the velocity would need the following inputs:

1. a start time
2. an end time
3. the number of rectangles
4. initial velocity

And it would have a single output:

1. The final velocity

```
[Vf] = IntegrateAcceleration(Nrect,ts,te,Vi)
```


Exercises

Modifying your numerical integration program to use functions is part of today's exercises.