PROGRAMMING SESSIONS 03

Computation: Control flow and Functions

Rafael Camacho

PROGRAM

Basic concepts

- Computation
- Control flow
- Managing complexity - abstraction
- Functions

Flow control in MATLAB

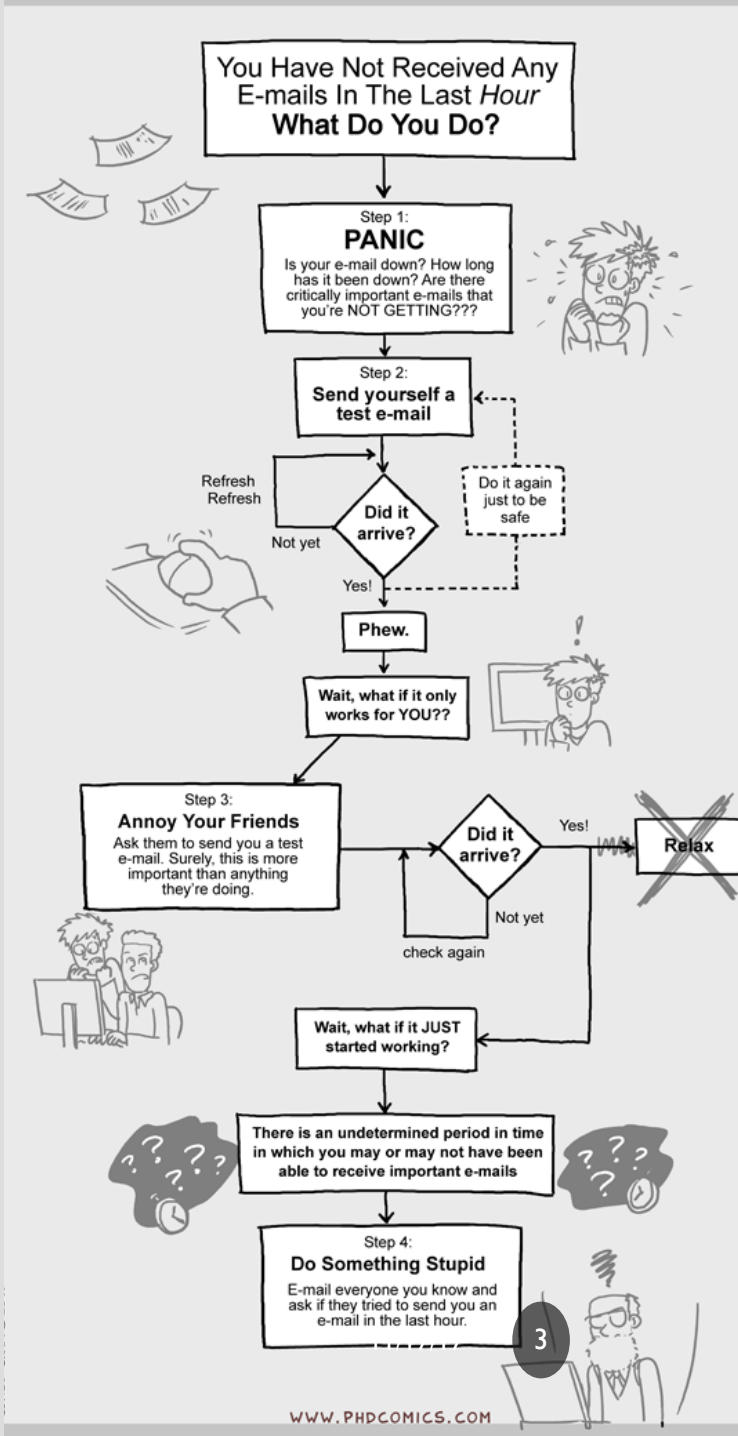
- Selection
  - If
  - Switch
- Iteration
  - While
  - For

Functions

- Why are functions important
- How to create a MATLAB function

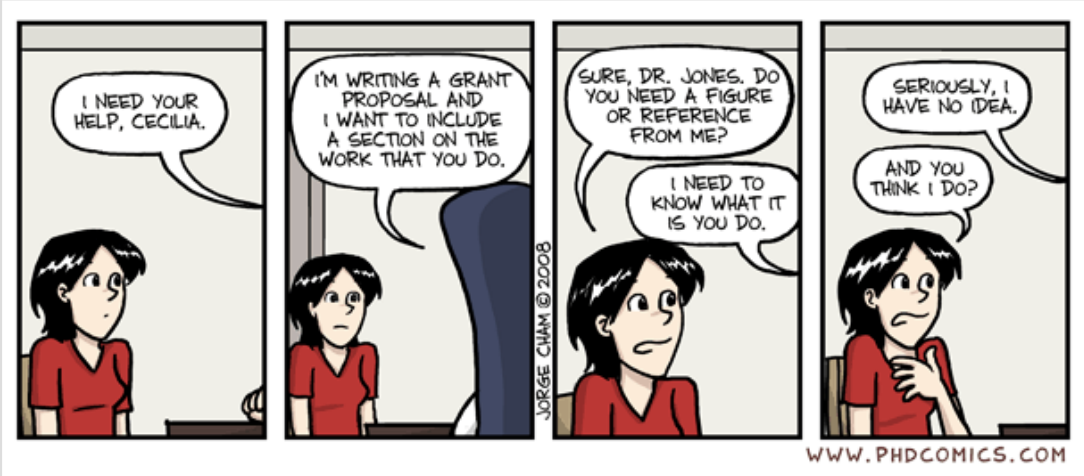
BASIC CONCEPTS

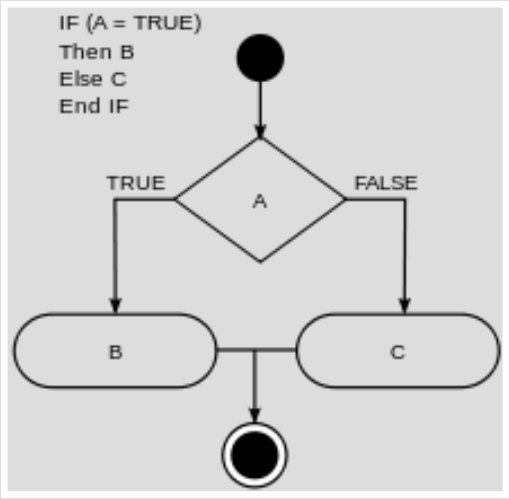
- **Computation**: refers to the act of producing some outputs based on some inputs.
- **Control flow:** refers to the order in which instructions are executed in a program. Among the different control flow statements we have: if, switch, while, for.



BASIC CONCEPTS

- The most important task you will have to tackle during programming is **managing complexity.** If you want to solve a very complex problem your main tool is to break such a problem into many little ones that are easier to handle (**divide and conquer**).
- **Abstraction**: hide details that we do not need in a convenient container, e.g. a **function**.
- A **function** is a named sequence of statement.





FLOW CONTROL IN MATLAB

One of the most basic decisions a program has to do is to select among alternatives.

An if-statement will execute a block of code if a condition is true.

MATLAB includes the elseif syntax. Beware! Does my code on the left behave as you would expect.

```
a = 1;

if a < 5
    disp('Smaller than 5!')
elseif a < 2
    disp('Smaller than 2!')
else
    disp('Larger than 5!')
end
```

FLOW CONTROL IN MATLAB

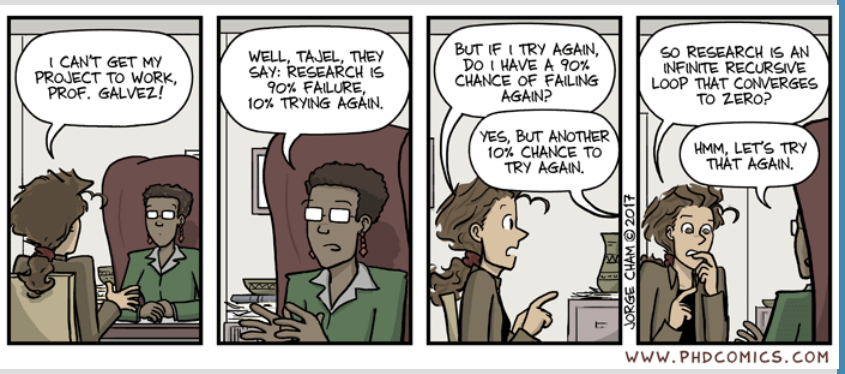
One of the most basic decisions a program has to do is to select among alternatives.

A switch-statement is designed for a very specific and common type of comparison, checking if a value is equal to a constant.

```
n = input('Enter a integer between 0-9: ');
n = uint16(n);
switch n
  case {2, 3}
    disp('You almost won! Good luck next time!')
  case 4
    disp('You won! AWESOME!')
  case {5,6}
    disp('You almost won! Good luck next time!')
  otherwise
    disp('Not even close! Good luck next time')
end
```

FLOW CONTROL IN MATLAB

```
n = input('Enter a number: ');
val = n;
go = isfinite(val);
counter = 0;
while go
    counter = counter + 1;
    val = val^2;
    go = isfinite(val);
end
fprintf('We had to consecutively square ...
```



Another very common thing that can happen when doing a computation is to repeat something several times. This repetition of the same block of statements is called **iteration**.

A while-loop will repeat the evaluation of some statement(s) as long as a condition is true.

Beware! Ensure that you do not generate an infinite loop by mistake.

FLOW CONTROL IN MATLAB

Another very common thing that can happen when doing a computation is to repeat something several times. This repetition of the same block of statements is called iteration.

Iterating over a list of elements is so common that most programing languages have a special syntax for it – the for-loop.

A for-loop is very similar to a while-loop except that this time we define how many times we will iterate before hand.

```
x = -10:1:10;
y = zeros(size(x));
for idx = 1:length(x)
    i_val = x(idx);
    y(idx) = i_val^2;
end
figure(1)
plot(x,y)
xlabel('x values')
ylabel('y values')
title('y = square(x)')
```

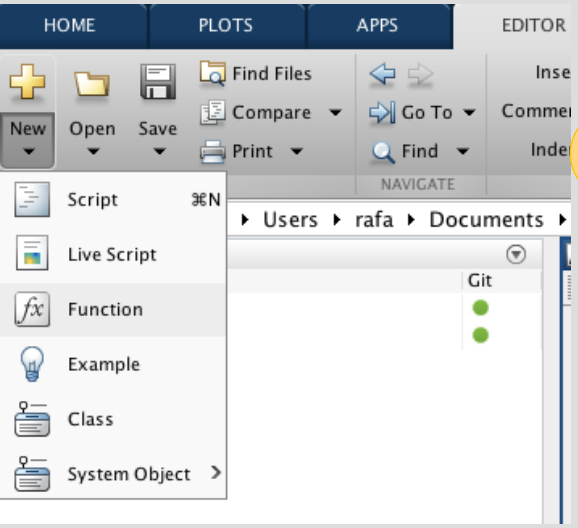
FUNCTIONS
WHY?

Functions are very useful when we want to:

- Divide the logic of a program into clear separate sub-calculations.
- Make the program easier to read.
- Make it possible to re-use a functionality.
- Make testing the program easier.

HOW TO CREATE A FUNCTION IN MATLAB

1



2

```
function [outputArg1,outputArg2] = untitled(inputArg1,inputArg2)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
outputArg1 = inputArg1;
outputArg2 = inputArg2;
end
```

3

```
function [out_val] = my_mean(vec)
%my_mean calculates the mean of all values in a vector.
%   We will explain this later

% check for precontidions
assert(isvector(vec),'input must be a vector')
sum_val = sum(vec);
n_elements = length(vec);
out_val = sum_val/n_elements;

end
```

ASSIGNMENT

The task for this session is to go back to project 01 step 01 and:

- Come up with a while-loop and for-loop implementation that calculates the Mandelbrot set. Basically we will solve together the last assignment.
- Save both implementations into a function file. Use a switch statement to let the user choose which implementation he wants to use, the for- or while-loop.
- As an extra bonus look into the 'tic' and 'toc' functions of MATLAB. What do they do? Use them to check which implementation is the fastest.