Computer Algebra and Technical Computing (MTH1006)

B. Vorselaars
bvorselaars@lincoln.ac.uk

School of Mathematics and Physics, University of Lincoln

Today

- Typical errors in a) functions and b) `while` loop
- Folders
- Programming practices

Recap

Consider the following problem

- Calculate the sum

$$S_m = \sum_{k=1}^{m} k$$

where m is an integer such that the sum just exceeds N, i.e. ,$S_m > N$ but $S_{m-1} \leq N$. Use 1) a script and 2) a function.

While example in script

Determine the minimum m, such that the sum of all natural numbers ranging from 1 to m exceeds N.

- `sum_exceeds_script` for $N = 1000$

```
N=1000;
m=0;
S=0;
while S<=N
  m=m+1;
  S=S+m;
end
m

>> sum_exceeds_script
m =
     45
```

Recap

- Solution while example in function

```
function m=sum_exceeds(N)
m=0;
S=0;
while S<=N
   m=m+1;
   S=S+m;
end
```

- `N`: input argument
- `m`: output argument
- `sum_exceeds`: function name, has to be the same as name of file.
- `function`: keyword indicating that the file is a function and not a script

Calling the function

```
>> x=sum_exceeds(3)
x =
     3
```

Indeed, for $m = 3$ the sum $\sum_{k=1}^{3} k = 6 > 3$

Common mistakes

- Forgetting the *output* argument

```
function sum_exceeds(N) % WRONG: 'm='
    omitted
m=0;
S=0;
while S<=N
  m=m+1;
  S=S+m;
end

>> x=sum_exceeds(3)
Error using sum_exceeds
Too many output arguments.
```

Common mistakes

- Both input *argument* (within parenthesis after function name) and assignment to a variable using the same name.

```
function m=sum_exceeds(N)
N=1; % WRONG: will overwrite input
    argument
m=0;
S=0;
while S<=N
  m=m+1;
  S=S+m;
end

>> x=sum_exceeds(3)
x =
     2
```

Wrong, answer should be 3 instead of 2.

Common mistakes

- Having no input *argument* (within parenthesis after the function name), but just setting the variable inside the function.

```
function n=sum_exceeds
N=1; % WRONG: this is not an input
    argument, but just setting the
    variable.
m=0;
S=0;
while S<=N
  m=m+1;
  S=S+m;
end

>> x=sum_exceeds(3)
Error using sum_exceeds
Too many input arguments.
```

Recap: while loop and question dialog

- Quiz

```
result=questdlg('Is it the case that
    x^2=-1 for x=-i?');
while ~strcmp(result, 'Yes')  % 'strcmp'
    compares the string, true for identical
     result=questdlg('This is wrong, try
         again')
end
```

The line `result=questdlg('This is wrong, try again')` is repeated until the answer is correct.

Typical errors with `while`

- Off-by-one error. A special type of logic error.
  Example: $\sum_{k=0}^{n} k$

```
sum=0; n=3; k=0;
while k<n
      sum=sum+k;
      k=k+1;
end
disp(sum)
```

What is the bug?

Typical error

- Off-by-one error. A special type of logic error.
  Example: $\sum_{k=0}^{n} k$

```
sum=0; n=3; k=0;
while k<n
    sum=sum+k;
    k=k+1;
end
disp(sum)
```

What is the bug?

```
sum=0; n=3; k=0;
while k<=n
    sum=sum+k;
    k=k+1;
end
disp(sum)
```

Expression is off by one, because $<$ is used instead of $\leq$

Typical error

- Example: $\sum_{k=0}^{n} k$

```
sum=0; n=3; k=0;
while k<=n
    sum=sum+k;
end
disp(sum)
```

What is the bug? The loop is *infinite*, because k is not increased within the `while` statement. The following line should be added:

```
k=k+1;
```

Folders

Windows has folders and unix/Mac has directories. They are basically the same. Properties folders:

- Storing data files (Word document, Excel, programs, Matlab files, etc.)
- Storing folders
  - $\rightarrow$ An hierarchical structure is possible

Folder hierarchy example

- `MyMatlabFolder`
  - `session1`
  - `session3`
    - `slides`
    - `exercises`

      `diary_session3.txt`
  - `session4`
  - `session5`
    - `exercises`
    - `assignment`

      `plot_polynomial.m`

      `my_figure.jpg`

Special folder names

Special folder name abbreviations

- . is the current folder
- .. is the parent folder that contains the current folder (one upwards in the folder hierarchy)
- / is the root folder, the folder at the top of the hierarchy (both Windows and Unix)
- \ is an alternative name for the root folder, but only for Windows

Folder commands

- `ls`: list the contents in a folder demo

```
>> ls
session1    session2    session3    session4
```

- `dir`: Similar command, show the content of the folder (directory)

```
>> dir
.          ..         session1   session2
   session3   session4
```

Folder commands II

- `pwd`: present work directory. The current folder. demo

```
>> pwd
ans =
/Users/Bart/MyMatlabFolder
```

- `cd`: change directory

```
>> cd session1
>> pwd
ans =
/Users/Bart/MyMatlabFolder/session1

>> cd ..
>> pwd
ans =
/Users/Bart/MyMatlabFolder
```

- `chdir`: the same as `cd`. demo

Creating and removing folders

- `mkdir`: make directory

```
>> ls
session1    session2    session3    session4
>> mkdir session5
>> ls
session1    session2    session3    session4
      session5
```

- `rmdir`: remove directory (only works when the directory is empty)

```
>> rmdir session5
>> ls
session1    session2    session3    session4
```

demo

Example folder

```
>> cd session4
>> edit myscript
>> type myscript %command to show the
   contents of myscript.m
disp('This script displays one line')
>> myscript
This script displays one line
>> cd ..
>> myscript
Undefined function or variable 'myscript'.
```

Scripts can only be run if they are in the current folder?

Path I

Path: all folders where Matlab searches for `.m` script and function files.

- `path`: shows all folders that are searched:

```
>> path
     C:\Program
        Files\MATLAB\R2022a\toolbox\matlab\addor
     C:\Program
        Files\MATLAB\R2022a\toolbox\matlab\addor
       ⋮
```

- `addpath`: add a folder to the path

```
>> cd session4
>> addpath(pwd)
>> cd ..
>> myscript
This script displays one line
```

Path II

- `savepath`: save the new search path for further Matlab sessions

```
>> quit
>> myscript
Undefined function or variable 'myscript'.
>> cd session4
>> addpath(pwd)
>> savepath
>> quit
>> myscript
This script displays one line
```

Programming practices

Good programming practices makes life easier

- Readable code
  - Choose variable and function names wisely
  - *Comment* the code by using %
  - Indent the code. The editor can help: CTRL-A (select all), then do CTRL-I (indent)
- Bad example

```
function c=f(a,b)
c=a./b.^2;
```

Meaning?

- Good example

```
function r=bmi(mass,height)
%Returns the body mass index (BMI), given
   mass in kg and height in m.
r=mass./height.^2;
```

→ Both functions are exactly the same, but 2nd is understandable

Programming practices

Strategies:

- *Incremental* programming: test code after (almost) every line. If the code does not work any more, then you know it is in the last line. Really
- Use a test script, to test the program and/or function for known (simple) values. Last example: test it for $n = 1$, and $n = 2$: *test-driven* development.
- Use safe guards within the function itself.
- *Debugging* (to be discussed later)

Assertion for testing

- `assert(logical_expression,'error message if false')`
- Example

```
function [x,y]=polar_to_Cartesian(r,
    theta)
%Converts polar coordinates to Cartesian
    coordinates.
assert(all(r(:)>=0),'The radial
    coordinate should be a non-negative
    number');
x=r.*cos(theta);
y=r.*sin(theta);
end
```

Assertion for testing

- The line

```
assert(all(r(:)>=0),'The radial
    coordinate should be a non-negative
    number');
```

is equivalent to

```
if ~(all(r(:)>=0))
  error('The radial coordinate should be
     a non-negative number')
end
```

Testing the assertion

```
>> [x,y]=polar_to_Cartesian(-1,pi/4)
??? Error using ==> polar_to_Cartesian at 5
The radial coordinate should be a
   non-negative number

>> [x,y]=polar_to_Cartesian(1,pi/4)
x =
    0.7071
y =
    0.7071
```