Lecture 9

Programming in Matlab

Programming in Matlab is similar to other scripting languages, such as Python.

Loops

The syntax of a "for" loop is:

for variable = range

computation body

end

Example use of a "for" loop to sum up the numers from 1 to 10:

```
s=0;
for i=1:10
    s=s+i;
end
s
```

```
s = 55
```

Compare to using the build in sum function

```
s=sum(1:10)
```

```
s = 55
```

Another way to realize a loop is to use the while statement. The syntax is

while expression

computation body

end

```
s=0;
i=1;
while i<=10
    s=s+i;
    i=i+1;
end
s
```

```
s = 55
```

Example of using while: how many numbers do we need to add up until we cross the sum of s=1000

```
s=0;
i=0;
while s<1000
    s=s+i;
    i=i+1;
end
[i   s]
```

```
ans = 1×2
        46        1035
```

The while loop is exited in case the condition s<1000 is false at the point of the end statement. The calculations in detail at each "end" statement:

$$\begin{bmatrix} i & s \\ 0 & 0 \\ 1 & 0 \\ 2 & 1 \\ 3 & 3 \\ 4 & 6 \\ 5 & 10 \\ \dots & \dots \\ 45 & 990 \\ 46 & 1035 \end{bmatrix}$$

At the time the code execution reach the keyword **end**, the counter i was already increased by 1 and we get 46 instead. The count that corresponds to 1035 is really 45

```
i=sum(1:45)
```

```
i = 1035
```

Using **IF** and **BREAK** to exit the loop

We can use a conditional break to exit the loop in order to get the desired result

```
s=0;
i=0;
while true
    s=s+i;
    if s>1000
        break;
    end
    i=i+1;
end
[i   s]
```

```
ans = 1×2
        45        1035
```

Conditional statements

The syntex of the **if** statement is

if condition

elseif condition

else condition

end

For example the following code converts an array of letter grades to GPA numbers

```
grades = ["A","C","D","B","F","A","B"];
v=[];
for g = grades
    if g=="A"
        v=[v 4.0];
    elseif g=="B"
        v=[v 3.0];
    elseif g=="C"
        v=[v 2.0];
    elseif g=="D"
        v=[v 1.0];
    else
        v=[v 0.0];
    end
end
v
```

```
v = 1×7
     4     2     1     3     0     4     3
```

Note that we are able to iterate over the grade array in a loop by assigning each element to g without having to make use of an index counter. Also, we initially define the result value v as empty array v and then append to it using the [ ] syntax.

Sometime we need to access the index of the array. For example, if we want to find the index of all students that got a As?

```
grades = ["A","C","D","B","F","A","B"];
gradeAstudents=[];
for i=1:length(grades)
    if grades(i)=="A"
        gradeAstudents = [gradeAstudents i]
    end
end
```

```
gradeAstudents = 1
gradeAstudents = 1×2
     1     6
```

```
gradeAstudents
```

```
gradeAstudents = 1×2
     1     6
```

Data Structures: containers.Map

Besiders the capability to store data in a list through the use of an array using the [ ] syntax, Matlab provides a data dictionary container. It works like a dictionary where values can be stored that are being referenced by a unique key.

We can define the variable M to be an empty containers.Map using

```
M = containers.Map;
```

and then assign values. For example making a dictionary of department room numbers

```
M('Computer Engineering Technology')='V-633';
M('Civil Engineering Technology')='V-433';
M('Mechanical Engineering Technology')='V-526';
M('Entertainment Engineering Technology')='V-205';
```

and query by key

```
M('Computer Engineering Technology')
```

```
ans = 
'V-633'
```

One use of this type of data structure is to tally counts of records. For example, let's say we have a list of costumes for each visitor that called on your house on Hallooween and now intend to tally up the counts for each costume. The following code generates a "histogram" of costume counts by using a containers.Map that keeps track of counts vs costume type.

```
records = ["It","Witch","Spiderman","Unicorn","Witch","It","Spiderman","It","Witch","It", "It",
costumeCount = containers.Map;
for costume=records
    if isKey(costumeCount,costume)
        costumeCount(costume)=costumeCount(costume)+1;
    else
        costumeCount(costume)=1;
    end
end
keys(costumeCount)
```

```
ans = 1×4 cell array
'It'         'Spiderman'     'Unicorn'     'Witch'
```

```
values(costumeCount)
```

```
ans = 1×4 cell
```

	1	2	3	4
1	5	3	2	4

Task 1

Given a time-series of stock prices, compute the maximum possible profit one can get by buying and selling. Assume you start with zero money and zero shares. At each time step you can either own one or zero shares. At the last day you need to sell any share you might still own.

Everytime you buy a share for a price, its cost gets deducted from your checking account money.

Everytine you sell a share, the selling prices gets credited to your money account.

Unlike in the real market your program is allowed to look at the future prices to determine the best timings.

In the example below I made $5 only. Can you do better? Hint: Buy low and sell high!

```
price = [100 110 120 140 130 105 80 70 85 99 110 150 140 130 90 105];
money = 0;
shares = 0;
for i=1:length(price)
    if(i==1 && shares==0)
        shares = shares+1;
        money = -price(i);
    end
    if(i==length(price) && shares==1)
        shares = shares-1;
        money = money+price(i);
    end
end
shares
```

```
shares = 0
```

```
money
```

```
money = 5
```

Task 2

Modify the Halloween program above to return:

a)The type of costumes have an odd number of visitors.

b)The costume with the maximum number of visitors.