Lecture III

Few recommendations about you…

1. When you do not know something (for example what Kronecker product or the Pascal triangle are) search *first* in www.wikipedia.org, *then* ask to your collegues and just *finally* ask to me. When you will program most probably me and your collegues won't be there to help you…
2. When you do not know how to use a function before asking write "help nameofthefunction" (for example help mean) in the command window, read carefully the instructions, and try to understand the basic features of the function and how to adapt it to your problem.
3. When you get an error do not get panic, read what Matlab is telling you: usually it will try to help you. If you do not understand immediately the mistake try to decompose your expression in simple smaller parts.
4. Learning how to program is a trial-and-error process do not get frustrated it takes time to write a code.

And about the problem sets…

5. Each problem set contains more exercises than you can face in 30minutes; if you want to learn you should do the rest by yourself at home.
6. To make the class useful for all of you, exercises have different degree of complexity. Some are trivial some are harder and of course some are wrong: you are supposed to understand why and correct them.
7. Even if an exercise looks trivial try to do it anyway, I'm sure it will not be as trivial as you though [please search overconfidence and overestimation in www.wikipedia.org]

Problem Set I [For]

1. Using “for”, write a scripts that displays 10 random numbers distributed according a U[0,1].

```
for i=1:10
rand
end
```

2. Using “for”, write a script that collects 10 random numbers U[0,1] in a column vector. Can you do the same using “vectorization”?

```
for i=1:10
R(i,:)=rand
end
```

 R=rand(10,1)

3. Using “for” write a script that collects 10 random numbers U[0,1] in a row vector. Can you do the same using “vectorization”?

```
for i=1:10
R(:,i)=rand
end
```

 R=rand(1,10)

4. Compare the outputs and the time needed to run these two scripts

a)
```
tic
for i=1:1000
i
end
toc
```

b)
```
tic
for i=1:1000
    I(i)=i
end
toc
```

Do you notice any difference?

The first script just display the results, the second store them in a vector, therefore the second takes more time

5. Compare the outputs of script b) in previous exercise and the following script

c)
```
tic
for i=1:1000
    I(i)=i;
end
```

```
toc
```

Do you notice any difference?

The output is the same but in order to run the last script it takes (much) less time

6. Replicate outputs of points a), b) and c) of previous exercise using "vectorization". What's the difference in terms of execution time?

```
tic
1:1000
toc

tic
I=1:1000
toc

tic
I=1:1000;
Toc
```

the moral is: USE VECTORIZATION whenever you can, it's faster!

7. Write a function [replicate.m] that, using as inputs "start" and "last" returns two outputs: "R" that replicate, through a for cycle, the output of start:last, and "Nobs" the number of observations

```
function [Nobs R]=replicate(start,last)

for i=start:last
R(i)=i
end
Nobs=length(R)
```

8. Given a=randi([1 20],1,20) using both for-cycle and vectorization (whenever possible) do the followings

   a. Create column vector "e" that contains the even elements in a

```
e=a(2:2:20)'

e=[]
for i=2:2:20
e=[e; a(i)]
end
```

   b. Create row vector "e" that contains the even elements in a

```
e=a(2:2:20)

e=[]
for i=2:2:20
e=[e a(i)]
end
```

c. Create column vector “o” that contains the even elements in a

```
o=a(1:2:20)’

o=[]
for i=1:2:20
o=[e; a(i)]
end
```

9. Compute n! where n=9

```
F=9*8*7*6*5*4*3*2*1

F=[1]
for i=2:9
    F=[F*i]
end

for i=1:9
    F(i)=[i]
end
prod(F)

factorial(9)

prod(1:9)
```

10. Replicate the first 20 elements of Fibonacci Series: 1 2 3 5 8 13 21 34 55 89 144… hint: for i>=3 $x_i = x_{i-1} + x_{i-2}$

```
  S=[1 2]
  for i=3:20
      S(i)=S(i-1)+S(i-2)
  end
```

11. Compute a=$\sum_{t=1}^{N} \frac{C}{(1+r)^t}$ where N=20 C=10 and r=0.05. Do it by vectorization

```
for t=1:20
      a(t)=10/(1+0.05)^t
end
sum(a)

C=10*ones(1,20)
R=(1+0.05).^[1:20]
sum(C./R)
```

12. Compute a=$\sum_{t=1}^{N} \frac{C_i}{(1+r_i)^t}$ where N=5 C=[10 11 12 13 14] r=[0.05 0.05 0.06 0.07 0.08]

C=[10:14]; r=[0.05 0.05 0.06 0.07 0.08];

```
for t=1:5
      a(t)=C(t)/(1+r(t))^t
end
sum(a)
```

13. Compute $\sum_{\mathrm{i}=0}^{\infty} \mathrm{x}^{\mathrm{i}} \quad = 1 + x^1 + x^2 + x^3$ ... for which values of x is it true that $\sum_{\mathrm{i}=0}^{\infty} \mathrm{x}^{\mathrm{i}} = \frac{1}{1-\mathrm{x}}$? [recall that $x^0 = 1$]

Is it true for $x\epsilon(-1,1)$

```
x=0.4
X=[];
for i=0:10000
    X=[X x^i];
end
sum(X)
```

sum(x.^[0:1000])

14. Compute $T = \sum_{i=1}^{N} \sum_{t=1}^{T} R_{t,i}$ and $C = \sum_{t=1}^{T} \sum_{i=1}^{N} R_{i,t}$ where R is a TxN matrix of random normal returns: `R=normrnd(0,2,T,N); N=10; T=100;`

```
for i=1:N
      P(i)=sum(R(:,i));
end
sum(P)
```

```
for i=1:T
      PP(i)=sum(R(i,:));
end
sum(PP)
```

15. Using for cycle replicate I=eye(5)

```
for i=1:5
I(i,i)=1
end
```

16. Build a 100x1 vector of prices (use lognormal distribution), compute the series of gross returns $R_t = \frac{P_t - P_{t-1}}{P_{t-1}}$

```
P=lognrnd(0,1,100,1)
for t=2:100
    R1(t)=(P(t)-P(t-1))/P(t-1)
end
```

R2=(P(2:end)- `P(1:end-1))./ P(1:end-1)`

17. Do the same but with a 100x10 matrix of prices

```
P=lognrnd(0,1,100,10)
for t=2:100
R1(t,:)=(P(t,:)-P(t-1,:))./P(t-1,:);
end
```

```
R2=(P(2:end,:)- P(1:end-1,:))./ P(1:end-1,:);
```

18. Using a nested for (for within a for) replicate A=$\begin{matrix}1 & 1 & 1 & 1\\ 2 & 2 & 2 & 2\\ 3 & 3 & 3 & 3\end{matrix}$ and B=$\begin{matrix}1 & 2 & 3 & 4\\ 1 & 2 & 3 & 4\\ 1 & 2 & 3 & 4\end{matrix}$

```
 for i=1:3
     for j=1:4
         X(i,j)=i
     end
end

for i=1:3
    for j=1:4
        X(i,j)=j
    end
end
```

19. Given a=randi([1 2],1,20) build a series W2 that contains the weighted average $w_t = \sum_{j=1}^{2} a_{t-j}$

```
W2=[]
for i=2:20
W2=[W2 mean(a(i-1:i))];
end
```

20. Generalize the previous exercise writing a function that accepts as inputs "TS" the original series, W the window of the weighetd average

```
[W]=waverage(TS,window)
for i=window:length(TS)
W=[W mean(a(i-window-1:window))]
end
```

21. Given an initial value of zero $x_o = 0$ Simulate 100 observations of an AR(1) process $x_t = \theta x_{t-1} + \varepsilon_t$ where $\varepsilon_t \sim N(0,1)$. Select three different values for θ: 0, 0.5, 0.99,1. After you did it write plot(1:100,x) in the command window and explain the results: for which value of θ the process is stationary?

```
x=0
theta=0
for t=2:100
    x(t)=theta*x(t-1)+randn
end
```

22. In the previous exercise we simulate one path, now simulate 1000 paths and store them in a matrix X(i,t) where each row contains a single path. Select two different values for $\theta: 0 \text{ and } 1$. Once you got the 1000x100 X matrix write plot(mean(X)) in the command window what happens when $\theta = 0$? What when is equal 1?

```
X=zeros(1000,1)
theta=1
for i=1:1000
for t=2:100
    X(i,t)=theta*X(i,t-1)+randn;
end
end
```

23. Run a for cycle of 20 iterations. At each iteration extract a random integer number $x \sim U(1,10)$ build a row vector 1:x and put it in a Matrix that stores all of them. Since row vectors will have different size to put them in a 20x10 matrix you have to "fill the gap" with zeros:

$$\begin{matrix} 1 & 2 & 3 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 0 & 0 & 0 \\ 1 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

```
for i=1:20
x=randi([1 10],1)
X=[X; 1:x zeros(1,10-x)]
end
```

Problem Set II [If]

1. Given x=rand write a script that displays ‘x is positive’ if x is strictly bigger than 0 and do nothing otherwise

```
x=rand
if x>0
    'x is positive'
end
```

2. Given x=rand write a script that displays ‘x is positive’ if x is strictly bigger than 0 and ‘x is negative’ if x less or equal 0

```
x=rand
if x>0
    'x is positive'
else
    'x is negative'
end
```

3. Given x=rand write a script that displays ‘x is positive’ if x is strictly bigger than 0 , ‘x is negative’ if x strictly less than 0 and ‘x is equal 0’ if x is exactly 0

```
x=rand
if x>0
    'x is positive'
elseif x<0
    'x is negative'
else
    'x is zero'
end
```

4. Given x= create an if structure that entails 4 cases ‘x is positive and strictly bigger than 0.5’, ‘x is positive and less than 0.5’, ‘x is negative and strictly less than -0.5, ‘x is negative and bigger than -0.5’

```
if x>0
    if x>0.5
        'x is positive and strictly bigger than 0.5'
    else
        'x is positive and less than 0.5'
    end
else
    if x<-0.5
       'x is negative and strictly less than -0.5'
    else
       'x is negative and bigger than -0.5'
    end
end
```

5. Try this code for x=0.9,x=0.2,x=-5, x=-0.3

```
if x>0
     'x is positive and less than 0.5'
elseif x>0.5
     'x is positive and strictly bigger than 0.5'
```

```
elseif x<-0.5
    'x is negative and strictly less than -0.5'
else   x<=0
    'x is negative and bigger than -0.5'
end
```

Why it displays the wrong message for x=0.9 ? why is correct for x=0.2, x=-5 and x=-0.3?
Change the structure in such a way that it works correctly!

```
 if x>0.5
      'x is positive and strictly bigger than 0.5'
elseif x>0
      'x is positive and less than 0.5'
elseif x<-0.5
      'x is negative and strictly less than -0.5'
 else   x<=0
     'x is negative and bigger than -0.5'
 end
```

Problem Set III [if and for]

1. Using for cycle and if replicate I=eye(5)

```
for i=1:5
     for j=1:5
         if j==i
         X(i,j)=1
         else
         X(i,j)=0
         end
      end
 end
```

2. Form a time series of rand, create a vector I whose entries are 1 if x>0.5 and 0 otherwise. Count the fractions of 1s you have after 10,50,100,1000 iterations. Do it by vectorization

```
for i=1:1000
    x=rand;
    if x>0.5
        I(i)=1;
    else
        I(i)=0;
    end
end

[sum(I(1:10)) sum(I(1:50)) sum(I(1:100)) sum(I(1:1000))]./[10 50 100 1000]

I=rand(1000,1)
I(find(I>0.5))=1
I(find(I~=1))=0
[sum(I(1:10)) sum(I(1:50)) sum(I(1:100)) sum(I(1:1000))]./[10 50 100 1000]
```

3. Generate a series of Returns (i.e. normrnd(0,1,100,1)) using for (and/or vectorization) and if, implement the following technical analysis trading signals [Build a series “Buy” that has 0 if the condition is not satisfied an 1]

   a. Buy if Return is bigger than 0.5
   b. Buy if Return is less than 0.5
   c. Buy if Return is less than 0.5 or bigger than 0.5
   d. Buy if Return is bigger than the average of the 5 past returns
   e. Buy if Return is bigger than the previous max

```
 R=normrnd(0,1,100,1)

 for i=1:length(R)


     if R(i)>0.5
         Buya(i)=1
     else
         Buya(i)=0
     end
```

```
    if R(i)<-0.5
         Buyb(i)=1
    else
         Buyb(i)=0
    end

    if  (Buya(i)==1 | Buyb(i)==1)
         Buyc(i)=1
    else
         Buyc(i)=0
    end

    if i>5
    if R(i)>mean(R(i-5:i))
        Buyd(i)=1
    else
        Buyd(i)=0
    end
    end

    if R(i)>max(R(1:i))
        Buye(i)=1
    else
        Buye(i)=0
    end
end
```