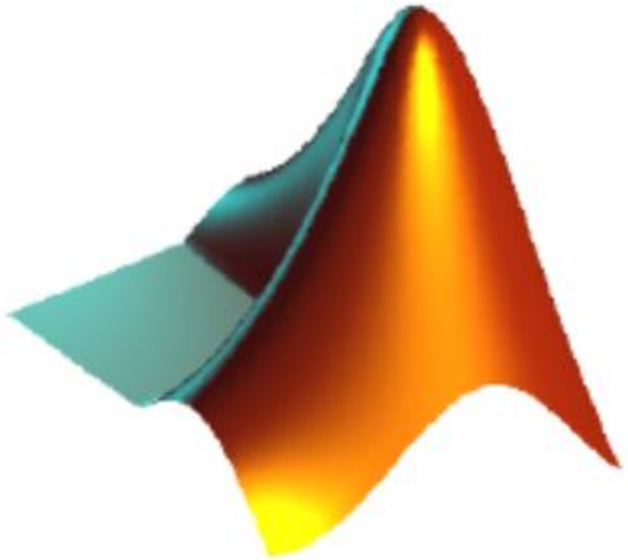
Programming in Matlab
LOOPS
(3)



LOOP TYPES

Loop Type	Description
while loop	Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
for loop	Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
nested loops	You can use one or more loops inside any another loop.

The while Loop

- The while loop repeatedly executes statements while condition is true.
- The while loop repeatedly executes program statement(s) as long as the expression remains true.
- An expression is true when the result is nonempty **and** contains all nonzero elements (logical or real numeric). Otherwise, the expression is false.
- **Syntax**

```
while <expression>
   <statements>
end
```

The while Loop Example

- Create a script file loop1.m and type the following code:

```
a = 10;
% while loop execution
while( a < 20 )
    fprintf('value of a: %d\n', a);
    a = a + 1;
end
```

- Run script loop1.m, the result will be:

```
>>loop1.m value of a: 10
          value of a: 11
          value of a: 12
          value of a: 13
          value of a: 14
          value of a: 15
          value of a: 16
          value of a: 17
          value of a: 18
          value of a: 19
```

The for Loop

- A **for** loop is a **repetition control structure** that allows you to efficiently write a **loop** that needs to **execute** a **specific number of times.**

- **Syntax**

```
for index = values
   <program statements>
   ...
end
```

The for Loop (Cont.)

values has one of the following forms:

Format	Description
initval:endval	increments the index variable from *initval* to *endval* by 1, and repeats execution of *program statements* until *index* is greater than *endval*.
initval:step:endval	increments *index* by the value step on each iteration, or decrements when step is negative.
valArray	creates a column vector *index* from subsequent columns of array*valArray* on each iteration. For example, on the first iteration, index = valArray(:,1). The loop executes for a maximum of n times, where n is the number of columns of *valArray*, given by numel(valArray, 1, :). The input *valArray* can be of any MATLAB data type, including a string, cell array, or struct.

The for Loop: Example

- Create a script file loop2.m and type the following code:

```
for a = 10:20
   fprintf('value of a: %d\n', a);
end
```

- Run script loop2.m, the result will be:

```
>>loop2.m value of a: 10
          value of a: 11
          value of a: 12
          value of a: 13
          value of a: 14
          value of a: 15
          value of a: 16
          value of a: 17
          value of a: 18
          value of a: 19
          value of a: 20
```

The for Loop: Example2

- Create a script file loop3.m and type the following code:

```
for a = 1.0: -0.1: 0.0
    disp(a)
end
```

- Run script loop3.m, the result will be:

```
>>loop3.m 1
          0.9000
          0.8000
          0.7000
          0.6000
          0.5000
          0.4000
          0.3000
          0.2000
          0.1000
          0
```

The for Loop: Example3

- Create a script file loop4.m and type the following code:

```
for a = [24,18,17,23,28]
    disp(a)
end
```

- Run script loop4.m, the result will be:

```
>>loop4.m 24
          18
          17
          23
          28
```

Quiz 1

- Let x is a row vector, x=[1, 3, 5, -.25, -.5];
- Compute f(x) as follows:

$$f(x) = \begin{cases} 1 + \cos(2\pi x), & |x| \leq \frac{1}{2} \\ 0, & |x| > \frac{1}{2}. \end{cases}$$

Quiz 1: Solution

$$f(x) = \begin{cases} 1 + \cos(2\pi x), & |x| \le \frac{1}{2} \\ 0, & |x| > \frac{1}{2}. \end{cases}$$

```
for j = 1:length(x)
  if abs(x(j)) <= 0.5
     f(j) = 1 + cos(2*pi*x(j));
  end
end
```

Quiz 1: Another Solution(without loops)

```
>>x=[1, 3, 5, -.25, -.5];

>>f=zeros(1, length(x));

>>r=find(abs(x)<=0.5);

>>f(r)=1+cos(2*pi*x(r))      % f([4,5])=1+cos(2*pi*x([4, 5]))
```

$$f(x) = \begin{cases} 1+\cos(2\pi x), & |x| \le \frac{1}{2} \\ 0, & |x| > \frac{1}{2}. \end{cases}$$

Quiz 1: Solve the following problems

Problem 1:

Check whether the value 5 exist in an array or not. If yes, return its index (1-D)

Problem 2:

Return the indices of all the occurrence of specific value (e.g., 7)in an array(1-D)

Problem 3:

Find and replace the value 8 with the value 9 from an array(1-D)

Problem 4:

Sort array in ascending order (1-D)

Problem 3 solution

```
>>x=[1 2 5 3 5 6 7 5 3 4 5]; with=9; value=5;
>>x=findReplace(x, value, with)

function x=findReplace(x, value, with)
   for i=1:length(x)
   r=find(x==value);
   x(r)=with;
   end
end
```

The Nested Loop

- **Nested Loops:** use one **loop inside another loop**.

- **Syntax**

```
for m = 1:j
  for n = 1:k
  <statements>;
  end
end
```

```
while <expression1>
while <expression2>
<statements>
end
end
```

The Nested Loop: Example

- Use a nested for loop to display all the prime numbers from 1 to 100.
- Create a script file loop5.m and type the following code:

```
for i=2:100
flag=0;
  for j=2:i-1
     if(~mod(i,j))
        flag=1;
        break; % if factor found, not prime
      end
    end

     if(flag==0)
         fprintf('%d is prime\n', i);
   end
end
```

The Nested Loop: Example results

- Run script loop5.m. the result will be:

```
>>loop5  2 is prime
         3 is prime
         5 is prime
         7 is prime
         11 is prime
         13 is prime
         17 is prime
         19 is prime
         23 is prime
         29 is prime
         31 is prime
         37 is prime
         41 is prime
         43 is prime
         47 is prime
             .
             .
             .
```

Quiz 2

- Suppose x is a column vector and you want to compute a matrix D such that $d_{ij} = x_i - x_j$.

Quiz 2: Solution (Nested Loops)

```
n = length(x);
D = zeros(n);          % initialization
for j = 1:n
    for i = 1:n
        D(i, j) = x(i) - x(j);
    end
end
```

The Loop Control statements

- **Loop control statements change execution from** its **normal sequence**.

- When **execution leaves** a **scope**, all automatic **objects** that were **created in** that **scope** are **destroyed**.

Control Statement	Description
break statement	Terminates the loop statement and transfers execution to the statement immediately following the loop.
continue statement	Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

The break statement

- The **break** statement **terminates execution** of **for or while loop.**
- **Statements in** the **loop** that appear **after** the **break** statement are **not executed.**

- In **nested loops**, **break exits** only from the **loop** in **which it occurs**. **Control passes** to the **statement following** the **end** of **that loop**.

The break statement Example

- Create a script file loop6.m and type the following code:

```
a = 10;
% while loop execution
while (a < 20 )
fprintf('value of a: %d\n', a);
a = a+1;
if( a > 15)
% terminate the loop using break statement
break;
end
end
```

- Run script loop6.m, the result will be:

```
>>loop6.m  value of a: 10
           value of a: 11
           value of a: 12
           value of a: 13
           value of a: 14
           value of a: 15
```

The continue statement

- The **continue** statement is used for **passing control** to **next iteration** of **for or while loop**.

Example:

- Create a script file loop7.m and type the following code:

```
a = 13;
%while loop execution
while a < 18
if a == 15
% skip the iteration
a = a + 1;
continue;
end
fprintf('value of a: %d\n', a);
a = a + 1;
end
```

- Run script loop7.m, the result

```
>>loop7.m value of a: 13
          value of a: 14
          value of a: 16
          value of a: 17
```