



INTRODUÇÃO AO MATLAB

SEATEL 2021

LEONARDO BARBOSA DA SILVA

A grayscale background image showing a person's hands typing on a laptop keyboard. The laptop screen displays CSS code, including lines for 'em.mail' and 'em.phone' with properties like 'background', 'display', 'width', 'height', 'float', and 'margin'.

MISSÃO DO CURSO:

**MOstrar QUE NÃO PRECISA SER
PROGRAMADOR PARA FAZER CÓDIGOS
EM MATLAB!**

AGENDA

- APRESENTAÇÃO DO PROGRAMA
- CONCEITOS BÁSICOS DE PROGRAMAÇÃO
- CRIAÇÃO DE SCRIPTS E FUNÇÕES CUSTOMIZADAS
- OPERAÇÕES COM VETORES E MATRIZES
- GERAÇÃO E FILTRAGEM DE SINAIS
- CRIAÇÃO DE GRÁFICOS
- TRABALHAR COM DADOS .TXT E .CSV
- EXPORTAR PROJETOS COMO RELATÓRIOS
- EXEMPLOS DE PROJETOS EM MATLAB NO ESCOPO DA GRADUAÇÃO

AULA 01

AULA 02

APRESENTAÇÃO DO PROGRAMA

- MATLAB é uma plataforma de programação e computação numérica usada em análise de dados, desenvolvimento de algoritmos e criação de modelos
- Neste curso, será utilizada a versão 2016b
- Linguagem de programação própria e de alto nível, similar a C/C++
- Amplamente usada por engenheiros e cientistas em áreas como:
 - Processamento Digital de Sinais
 - Sistemas de Controle
 - Sistemas de Comunicação
 - Sistemas Elétricos de Potência
 - Processos Estocásticos
 - Inteligência Artificial
- Sua documentação pode ser encontrada em:
<https://www.mathworks.com/help/matlab/>

APRESENTAÇÃO DO PROGRAMA

- **ALGUMAS ALTERNATIVAS AO MATLAB:**



*Programa gratuito com interface e funções equivalentes
<https://www.gnu.org/software/octave/download>



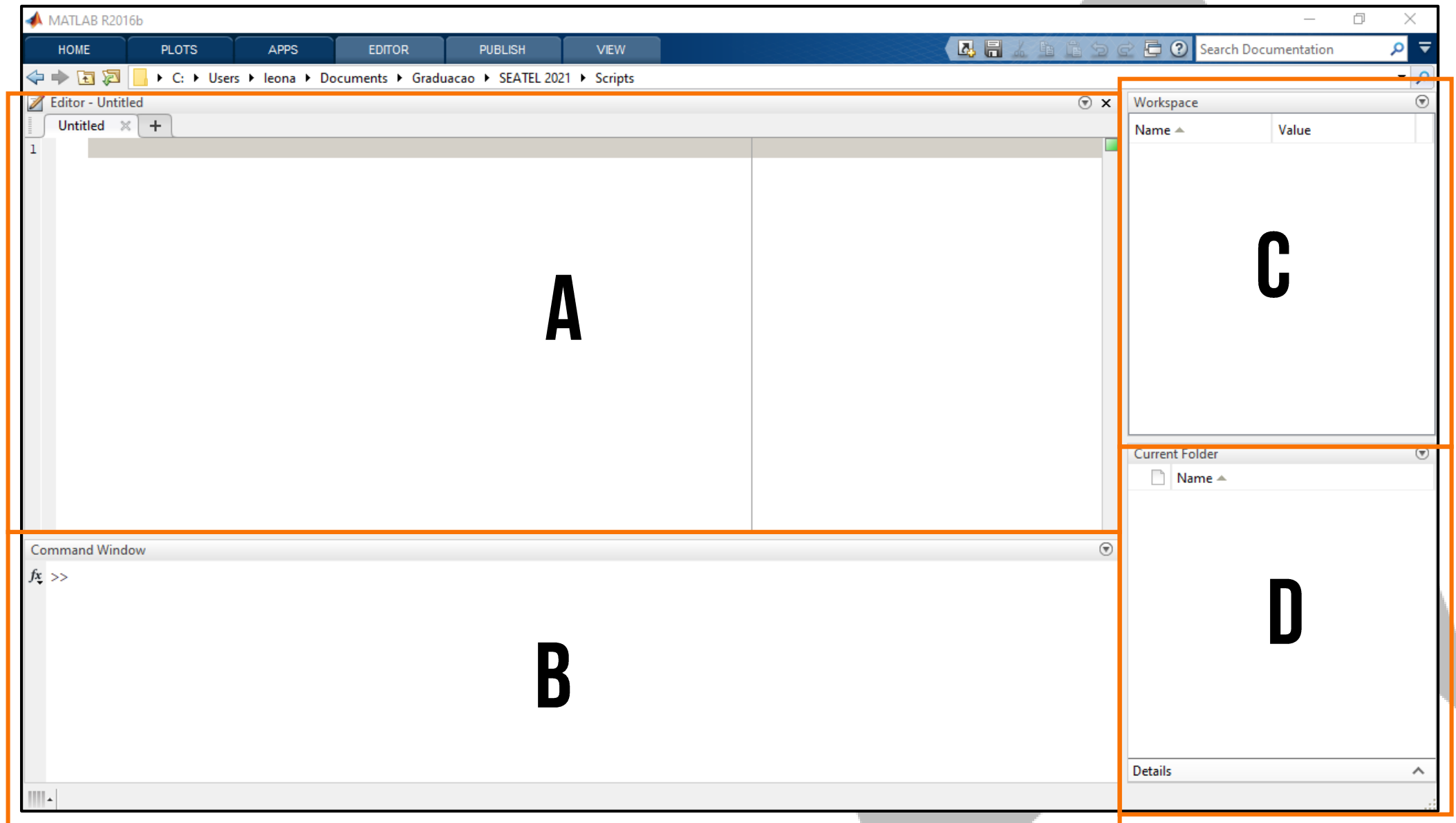
*Linguagem diferente, mas que produz resultados similares
<https://www.python.org/downloads/>

No formato jupyter notebook (login e senha do moodle):
<https://edison.eletrica.ufpr.br/jupyter/hub/login>

APRESENTAÇÃO DO PROGRAMA

- **PRINCIPAIS JANELAS DA INTERFACE:**

- A. **Editor (editor):** janela de elaboração de algoritmos repetíveis
- B. **Janela de comandos (command window):** entrada individual de comandos (statements) para operações mais simples, debug ou para chamar funções e ferramentas
- C. **Diretório atual (current folder):** mostra em qual diretório as operações estão sendo aplicadas
- D. **Área de trabalho (workspace):** mostra as variáveis criadas, seus tipos e valores



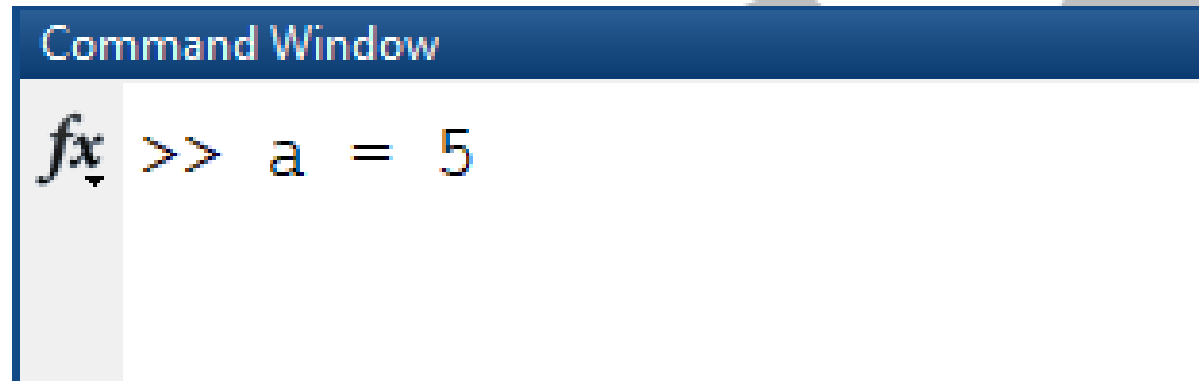
CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- **TIPOS (CLASSES) DE VARIÁVEIS MAIS COMUNS:**
 - int (16 e 32) – Números inteiros positivos ou negativos
 - double – Números decimais positivos ou negativos. É a “variável padrão” do MATLAB
 - complex double – Números números complexos na forma retângular
 - char – Caractere único ou em vetor
 - string – Cadeia de caracteres
 - logical – Variável booleana
- **Obs:** em muitos casos não é necessário declarar o tipo ao criar uma variável, porém é importante conhecê-las para usar corretamente as funções

CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- **CRIANDO UMA VARIÁVEL:**

- Na janela de comando, dê um nome para a variável, seguida de um “ = ” e o valor:



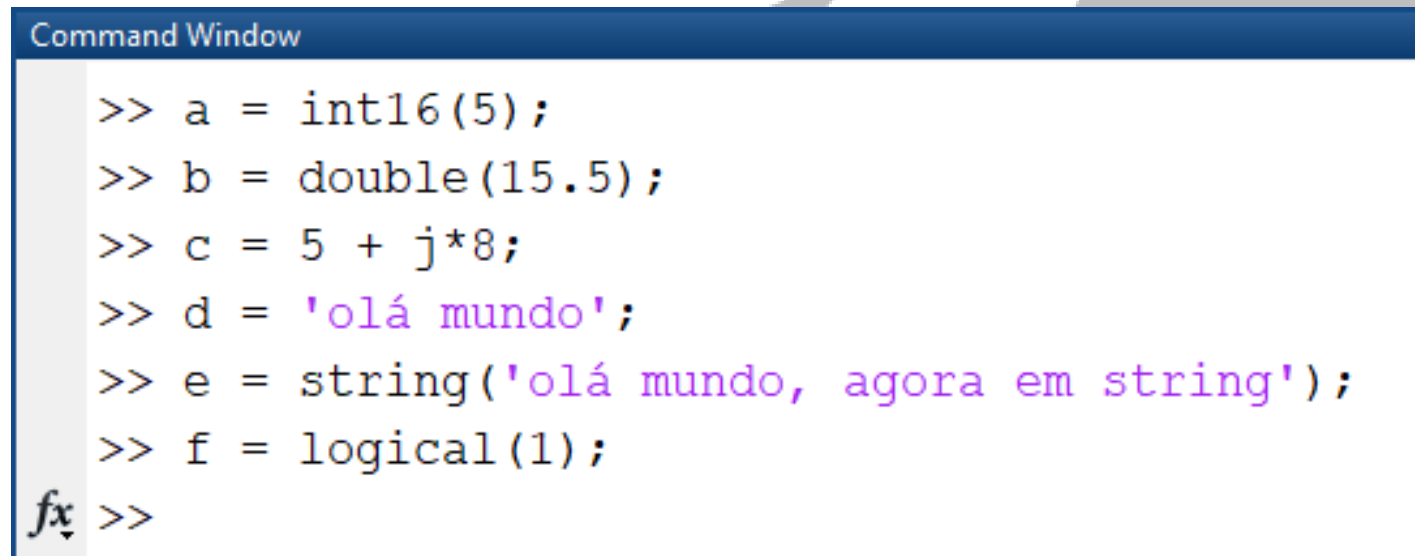
```
fx >> a = 5
```

- O programa diferencia letras maiúsculas, minúsculas e aceita underline (_). Para valores decimais, usar um “ . ”

CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- **CRIANDO UMA VARIÁVEL:**

- Para escolher o tipo da variável, usar as funções e notações da classe desejada

A screenshot of the MATLAB Command Window. The title bar is dark blue with the text "Command Window" in white. The main area is white and contains several lines of MATLAB code entered at the prompt ">>". The code defines variables a, b, c, d, e, and f with different data types. The string literals are highlighted in purple. At the bottom left, there is a small icon of a cursor and the text "fx" followed by ">>".

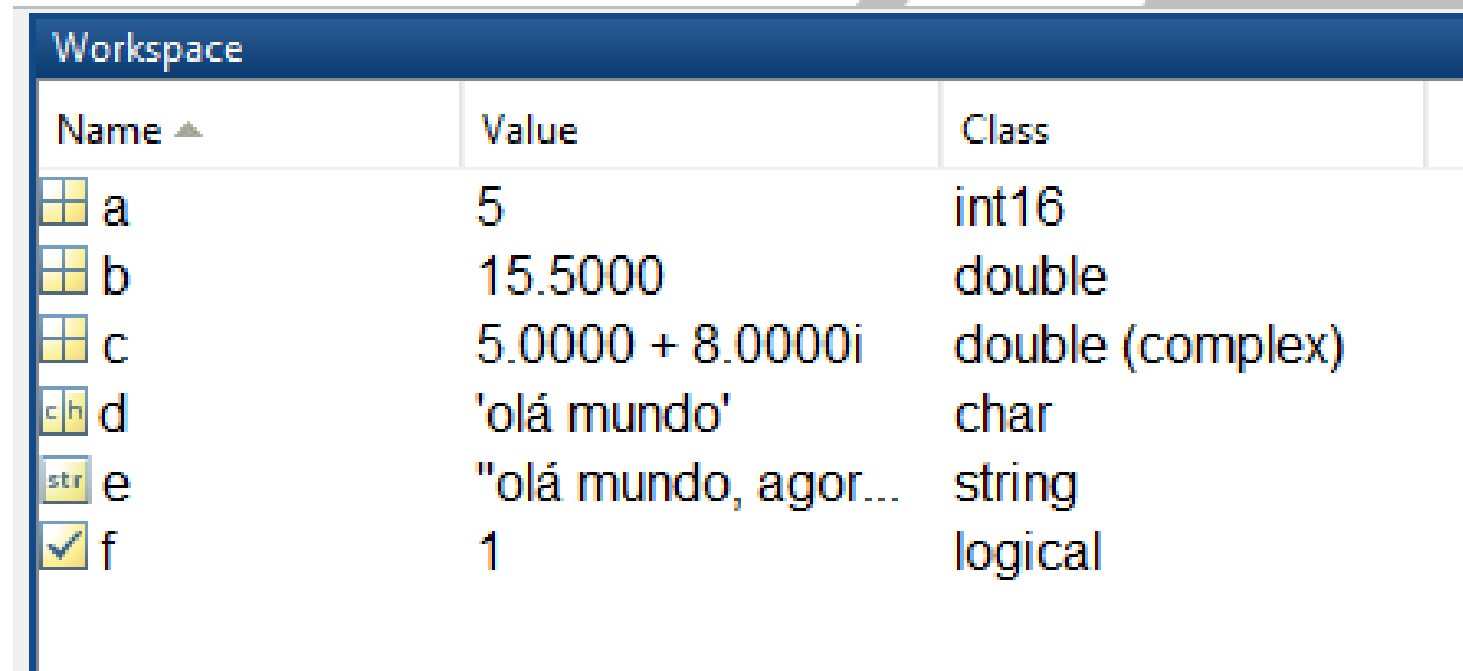
```
>> a = int16(5);  
>> b = double(15.5);  
>> c = 5 + j*8;  
>> d = 'olá mundo';  
>> e = string('olá mundo, agora em string');  
>> f = logical(1);  
fx >>
```

- Para conferir o tipo de uma variável, podemos usar os comandos “class()” e “is____()”







CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- **CRIANDO UMA VARIÁVEL:**

- Também podemos ver quais são as variáveis existentes na Workspace



The screenshot shows the MATLAB Workspace window. It contains a table with three columns: Name, Value, and Class. The variables listed are a, b, c, d, e, and f. Variable 'f' is selected, indicated by a checkmark icon in the Name column.

Name ▲	Value	Class
 a	5	int16
 b	15.5000	double
 c	5.0000 + 8.0000i	double (complex)
 d	'olá mundo'	char
 e	"olá mundo, agor..."	string
 <input checked="" type="checkbox"/> f	1	logical

- Ou então, usando os comandos “who” e “whos”

CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- **CRIANDO UMA VARIÁVEL:**

- Ou então, usando os comandos “who” e “whos”

```
>> who
```

```
Your variables are:
```

```
a b c d e f
```

```
>> whos
```

Name	Size	Bytes	Class	Attributes
a	1x1	2	int16	
b	1x1	8	double	
c	1x1	16	double	complex
d	1x9	18	char	
e	1x1	182	string	
f	1x1	1	logical	

CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- **CRIANDO UMA VARIÁVEL:**

- Podemos criar mais de uma variável se as separarmos por “ , ”

```
>> a = 3, b = 4
```

```
a =
```

```
3
```

```
b =
```

```
4
```

```
>> a = int16(12), b = double(15.5)
```

```
a =
```

```
int16
```

```
12
```

```
b =
```

```
15.500000000000000
```

CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- **CRIANDO UMA VARIÁVEL:**

- Vale lembrar que existem alguns nomes de variáveis que não são aceitos ou recomendados:
 - O nome da variável deve obrigatoriamente começar com uma letra
 - Não é recomendado usar nomes de variáveis e funções já reservados pelo programa, como por exemplo: “ans”, “pi”, “inf”, “i” e “j”

```
>> a = pi  
  
a =  
  
3.141592653589793
```

CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- COMANDOS ÚTEIS PARA INTERAGIR COM A JANELA DE COMANDO E A WORKSPACE

Comando	Função
clc	Apaga tudo que está na janela de comando
clear	Apaga todas as variáveis da Workspace

- **Obs:** Podemos apagar somente variáveis específicas se usarmos “clear nome_variavel”

CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- OPERAÇÕES MATEMÁTICAS:

Operação	Símbolo
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Potência	^

Command Window

```
>> 1 + 1;  
>> 2 - 5;  
>> 5.4 * 8;  
>> 15 / 5;  
>> 4^8;  
fx >>
```


CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- **ALGUNS OUTROS SÍMBOLOS ESPECIAIS:**

Símbolo	Função
,	Separa variáveis em uma mesma linha
;	Suprime a impressão na janela de comando
%	Começa um comentário
...	Permite estender um comando para mais de uma linha

CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- **OPERAÇÕES MATEMÁTICAS:**

- Combinando o que já aprendemos:

Command Window

```
>> a = 2;  
>> b = a + 5;  
>> c = b^2
```

c =

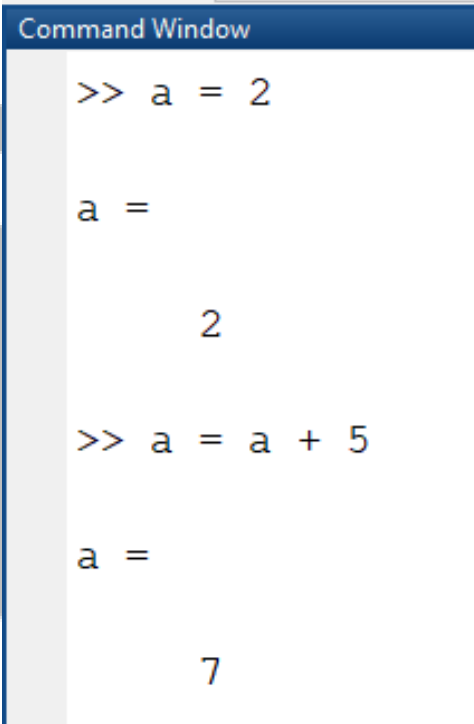
49

CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- **OPERAÇÕES MATEMÁTICAS:**

- Vale lembrar que o “ = ” não significa igualdade em programação, mas sim a operação de atribuição

- Isso significa que o seguinte comando é possível:



```
Command Window
>> a = 2
a =
    2
>> a = a + 5
a =
    7
```

CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- **OPERAÇÕES MATEMÁTICAS:**

- Algumas outras funções importantes

Operação	Símbolo	Função
Raíz quadrada	\sqrt{x}	sqrt(x)
Exponencial neperiana	e^x	exp(x)
Logaritmo neperiano	$\ln(x)$	log(x)
Logaritmo base 10	$\log_{10}(x)$	log10(x)
Seno	$\sin(x)$	sin(x)
Cosseno	$\cos(x)$	cos(x)
Tangente	$\tan(x)$	tan(x)
Módulo	$ x $	abs(x)
Resto da divisão	$x \bmod y$	mod(x,y)

CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- **OPERAÇÕES MATEMÁTICAS:**

- Funções para operar com números imaginários

- $M\angle\theta = M * e^{j\theta} = a + jb$

Operação	Função
Módulo (M)	abs()
Ângulo (θ) em radianos	angle()
Parte real	real()
Parte imaginária	imag()

CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- **LÓGICA BÁSICA:**

Operação relacional	Símbolo
Maior	>
Menor	<
Maior ou igual	>=
Menor ou igual	<=
Igual a	==
Diferente de	!=

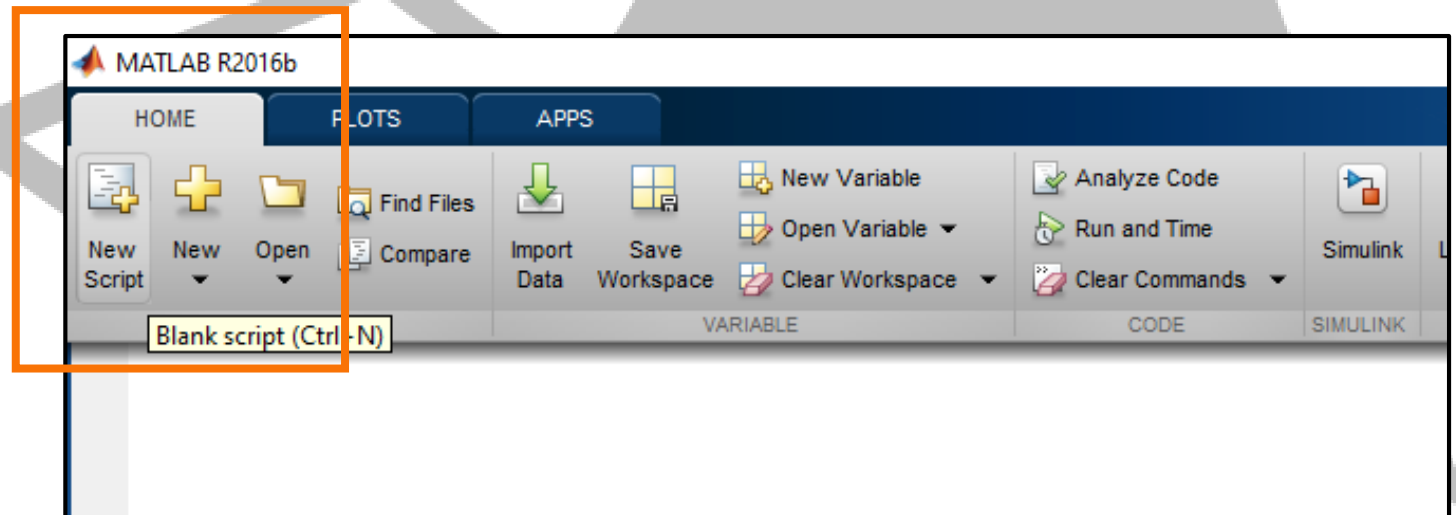
Operação lógica	Símbolo
Negação	~
E (and)	&&
Ou (or)	

Obs: And e or são operações também chamadas de curto-circuito

CRIAÇÃO DE SCRIPTS E FUNÇÕES CUSTOMIZADAS

- **ABRINDO A JANELA DO EDITOR**

- Para criar funções mais complexas e que possam ser executadas repetidas vezes, ao invés de utilizar a janela de comandos, opta-se por desenvolver um script
- Na barra superior, clicar na aba “Home” e em seguida em “New script”



CRIAÇÃO DE SCRIPTS E FUNÇÕES CUSTOMIZADAS

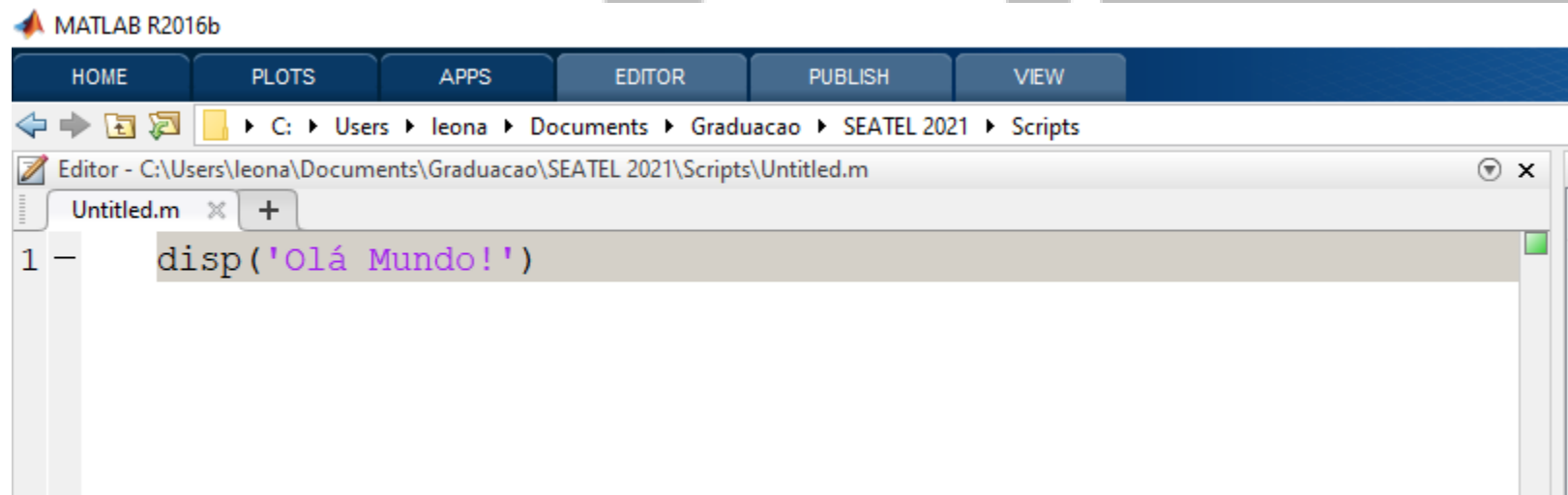
- **ABRINDO A JANELA DO EDITOR**

- Uma janela chamada “Editor” será criada e nela podemos criar nossos próprios scripts
- Os scripts do MATLAB são arquivos de extensão “.m” que quando executados no programa, realizam todos os comandos inseridos nele
- Como primeira demonstração, iremos fazer o clássico script do “Olá Mundo! ”

CRIAÇÃO DE SCRIPTS E FUNÇÕES CUSTOMIZADAS

- **OLÁ MUNDO**

- Neste primeiro script, usaremos uma nova função, a “disp()” que imprime uma variável na janela de comando
- Escreva a mensagem desejada entre aspas simples e dentro dos parênteses da função



CRIAÇÃO DE SCRIPTS E FUNÇÕES CUSTOMIZADAS

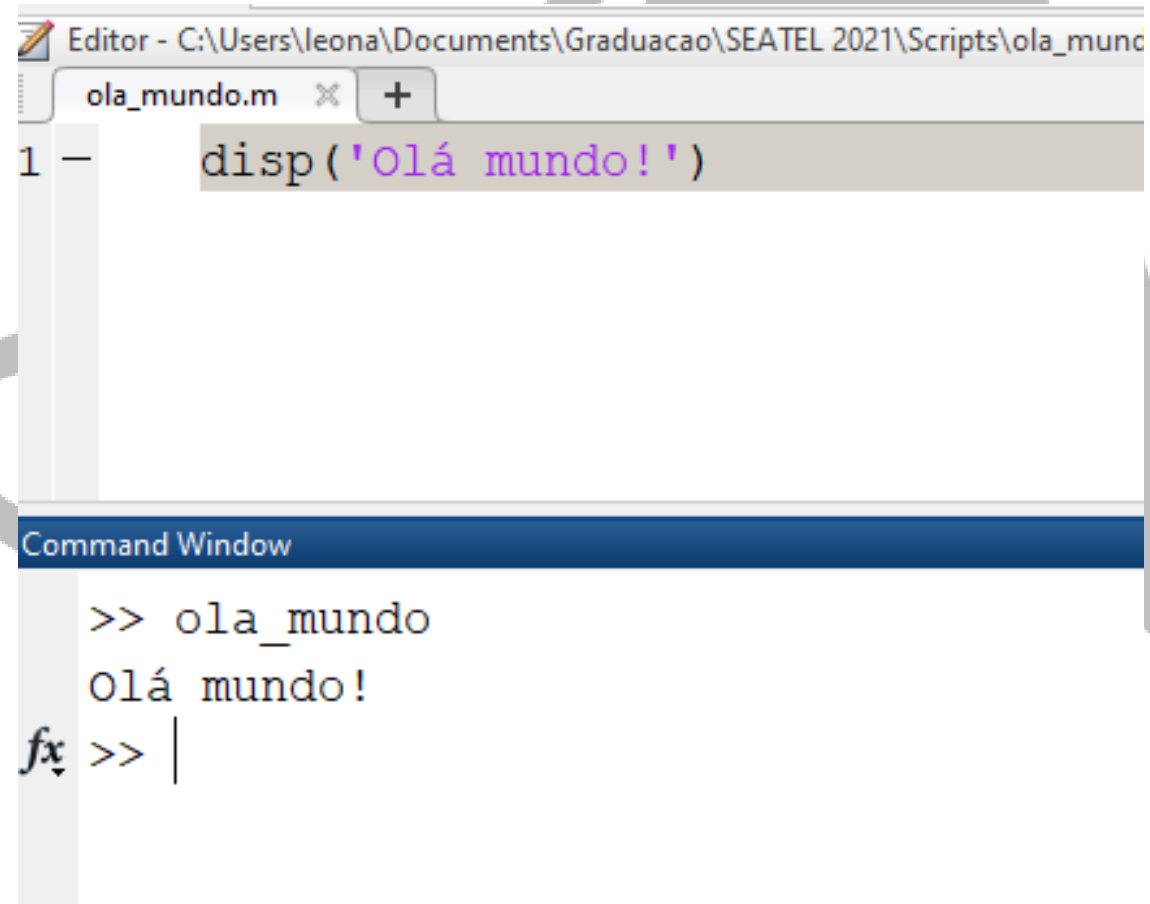
- **OLÁ MUNDO**

- Agora que já criamos um primeiro script, podemos salvá-lo. Isto pode ser feito com “Ctrl+s” ou indo na aba “Editor” da barra superior e clicando em “Save”
- Recomenda-se salvar todos os scripts de um mesmo projeto em uma mesma pasta
- Com o script salvo, agora podemos executá-lo digitando o nome do arquivo na janela de comando ou clicando em “Run” na aba “Editor”

CRIAÇÃO DE SCRIPTS E FUNÇÕES CUSTOMIZADAS

- **OLÁ MUNDO**

- O script se chama “ola_mundo”
- Assim, se digitarmos “ola_mundo” na janela de comando...
- A mensagem irá ser impressa



The screenshot shows a MATLAB environment. The top window is the 'Editor' with the file 'ola_mundo.m' open. It contains a single line of code: `disp('Olá mundo!')`. Below the editor is the 'Command Window'. It shows the command `>> ola_mundo` being entered, followed by the output `Olá mundo!`. The prompt `fx >> |` is visible at the bottom of the Command Window.

```
Editor - C:\Users\leona\Documents\Graduacao\SEATEL 2021\Scripts\ola_mundo.m
ola_mundo.m x +
1 - disp('Olá mundo!')

Command Window
>> ola_mundo
Olá mundo!
fx >> |
```

CRIAÇÃO DE SCRIPTS E FUNÇÕES CUSTOMIZADAS

- **RELEMBRANDO ESTES COMANDOS**

- Estes dois comandos são muito úteis na inicialização de scripts, pois em alguns casos não queremos que um resultado de uma execução afete a seguinte
- Isto será útil principalmente quando tratarmos de loops

Comando	Função
clc	Apaga tudo que está na janela de comando
clear	Apaga todas as variáveis da Workspace

- **Obs:** Podemos apagar somente variáveis específicas se usarmos “clear nome_variavel”

CRIAÇÃO DE SCRIPTS E FUNÇÕES CUSTOMIZADAS

- **ESTRUTURAS CONDICIONAIS:**

- Com o que aprendemos, podemos fazer as nossas primeiras estruturas de tomada de decisão com as funções if, else e elseif

```
if <condição A>  
    <comando A >  
elseif <condição B>  
    <comando B >  
else  
    <comando C >  
end
```

- Estas condições são determinadas pelo usuário e são compostas pelas diversas operações lógicas e relacionais vistas anteriormente
- **Obs:** são possíveis estruturas somente com “if” ou também “if-else” e “if-elseif”

CRIAÇÃO DE SCRIPTS E FUNÇÕES CUSTOMIZADAS

- **ESTRUTURAS CONDICIONAIS:**

- Como exemplo, podemos fazer uma função que pede para o usuário informar um número inteiro. Nosso script irá então dizer se o número digitado é par ou ímpar
- Para isso, vamos usar a função “input()” que pede para o usuário uma informação através da janela de comando
- Usaremos a estrutura condicional vista anteriormente para validar alguns erros que o usuário pode cometer

CRIAÇÃO DE SCRIPTS E FUNÇÕES CUSTOMIZADAS

```
1 clear
2 clc
3
4 disp('Digite um número inteiro positivo:')
5
6 x = input('x = ');
7
8 if isempty(x) || ischar(x) || x < 0 || mod(x,1) ~= 0
9     disp('Favor digitar um número inteiro positivo')
10 elseif mod(x,2) ~= 0
11     disp('O número informado é ímpar!')
12 else
13     disp('O número informado é par!')
14 end
```

CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- **LOOPS:**

- São estruturas que repetem um bloco de comandos enquanto uma condição lógica não é satisfeita.
- **Loop for:** realiza o bloco um número de vezes específico (de x até y, com iterador n)
- **Loop while:** enquanto a condição for verdadeira, executa o bloco

```
while <condição>  
    <bloco >  
end
```

```
for n = x:y  
    <bloco >  
end
```


CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- EXEMPLO DE LOOP FOR:

```
1 - clear
2 - clc
3
4 - a = 10;
5
6 - for n = 1:a
7 -     fprintf('loop na iteração %d \n', n)
8 - end
```

Obs: Podemos usar esta estrutura para varrer cada célula de uma tabela por exemplo

JANELA DE COMANDO

Command Window

```
loop na iteração 1
loop na iteração 2
loop na iteração 3
loop na iteração 4
loop na iteração 5
loop na iteração 6
loop na iteração 7
loop na iteração 8
loop na iteração 9
loop na iteração 10
```

fx >>

CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- EXEMPLO 2 DE LOOP FOR (INCREMENTO \neq 1):

```
1 - clear
2 - clc
3
4 - a = 10;
5
6 - for n = 1:2:a
7 -     fprintf('n tem valor %d \n', n)
8 - end
```

Obs: Podemos usar esta estrutura para varrer somente coordenadas pares ou ímpares de uma tabela por exemplo

JANELA DE COMANDO

Command Window

```
n tem valor 1
n tem valor 3
n tem valor 5
n tem valor 7
n tem valor 9
```

fx >>

CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- EXEMPLO DE LOOP WHILE:

```
1 - clear
2 - clc
3
4 - a = 15;
5
6 - while a >= 5
7 -     fprintf('a é: %d \n',a)
8 -     a = a - 2;
9 - end
10
11 - fprintf('Agora "a < 5", terminando com valor %d \n', a)
```

JANELA DE COMANDO

Command Window

a é: 15

a é: 13

a é: 11

a é: 9

a é: 7

a é: 5

Agora "a < 5", terminando com valor 3

fx >> |

Obs: Podemos usar esta estrutura para iterar uma função até o usuário informar um valor válido

CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- **ALGUNS CUIDADOS COM LOOPS:**

- Cuidado com a condição lógica do loop while, garantindo que ela realmente será falsa em algum momento. Do contrário, o loop executará “infinitamente”
- Loops longos ou “infinitos” podem apresentar alto custo computacional, que aumentam caso o bloco realize diversas tarefas em cada iteração
- A indentação não é obrigatória, porém é uma boa prática para legibilidade do código. Principalmente quando usamos loops e funções condicionais em conjunto
- Não esquecer de indicar com “end” onde é o fim do loop

CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- **CRIAÇÃO DE FUNÇÕES CUSTOMIZADAS:**

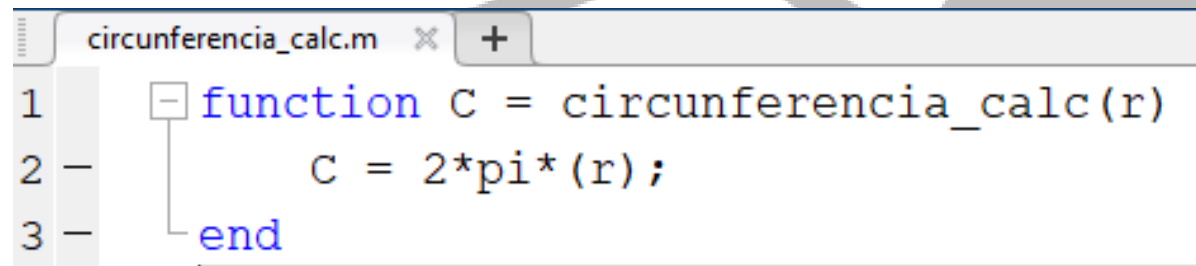
- Em programação, uma regra importante para legibilidade e desempenho do script é “Não Se repita” (do inglês Don’t Repeat Yourself, ou DRY)
- Assim, para casos em que uma mesma operação matemática é realizada em muitos trechos de um mesmo código, costuma-se definir uma função customizada

```
function saída = nome_funcao(argumentos)
    <bloco de comandos>
end
```

CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- **CRIAÇÃO DE FUNÇÕES CUSTOMIZADAS:**

- Exemplo: criar uma função para calcular a circunferência de um círculo dado um raio informado em metros:



```
circunferencia_calc.m  ✕  +  
1  function C = circunferencia_calc(r)  
2  -      C = 2*pi*(r);  
3  -  end
```

CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- CRIAÇÃO DE FUNÇÕES CUSTOMIZADAS:

- Se o nome do script e da função forem iguais, podemos inclusive usá-la em outros scripts

```
Editor - C:\Users\leona\Documents\Graduacao\SEATEL 2021\Scripts\teste_function.m
circunferencia_calc.m x teste_function.m x +
1 - clear
2 - clc
3 -
4 - for n = 1:10
5 -     c = circunferencia_calc(n);
6 -     fprintf('Nova circunferência: %f m^2 \n', c)
7 - end
```

```
Command Window
Nova circunferência: 6.283185 m^2
Nova circunferência: 12.566371 m^2
Nova circunferência: 18.849556 m^2
Nova circunferência: 25.132741 m^2
Nova circunferência: 31.415927 m^2
Nova circunferência: 37.699112 m^2
Nova circunferência: 43.982297 m^2
Nova circunferência: 50.265482 m^2
Nova circunferência: 56.548668 m^2
Nova circunferência: 62.831853 m^2
fx >>
```

CONCEITOS BÁSICOS DE PROGRAMAÇÃO

- **CRIAÇÃO DE FUNÇÕES CUSTOMIZADAS:**

- Caso deseja-se definir a função no mesmo script porém, isto deve ficar como parte final do script e os nomes devem ser diferentes

```
Editor - C:\Users\leona\Documents\Graduacao\SEATEL 2021\Scripts\teste_function_2.m
teste_function_2.m  x +
1 - a = 10;
2
3 - c = circunferencia_calc(2)
4
5 - function C = circunferencia_calc(r)
6 -     C = 2*pi*(r);
7 - end
```

```
Command Window
>> teste_function_2

c =

12.566370614359172
```


OPERAÇÕES COM VETORES E MATRIZES

- **CRIAÇÃO DE VETORES**

- Vetores são variáveis 1D de tamanho n
- No MATLAB, podemos criá-las das seguintes formas:

Tipo de vetor	Código
De a até b, de 1 em 1	<code>V = a:b</code>
De a até b, com incrementos de n	<code>W = a:n:b</code>
De a até b, com m elementos igualmente espaçados	<code>X = linspace(a,b,m)</code>
Elementos declarados explicitamente	<code>Y = [a b c d]</code>
Coluna de elementos declarados explicitamente	<code>Z = [a; b; c; d]</code>

OPERAÇÕES COM VETORES E MATRIZES

- **ALGUMAS OPERAÇÕES COM VETORES**

- Para acessar um elemento específico de um vetor:

```
Command Window
>> A = [1 2 5 9];
>> A(3)

ans =

    5
```

Obs: ao contrário da maioria das linguagens, em MATLAB o primeiro índice é o 1

OPERAÇÕES COM VETORES E MATRIZES

- **ALGUMAS OPERAÇÕES COM VETORES**

- Podemos acessar também múltiplos elementos de um vetor:

```
Command Window
>> A = linspace(1,15,29);
>> A(5:10)

ans =

Columns 1 through 3

3.0000000000000000    3.5000000000000000    4.0000000000000000

Columns 4 through 6

4.5000000000000000    5.0000000000000000    5.5000000000000000
```

OPERAÇÕES COM VETORES E MATRIZES

- **ALGUMAS OPERAÇÕES COM VETORES**

Operação	Código
Operações ponto-a-ponto	$A .* B$
	$A ./ B$
	$A .^ B$
Transposição	A'
Média	<code>mean(A)</code>
Desvio padrão	<code>std(A)</code>
Valor máximo	<code>max(A)</code>
Valor mínimo	<code>min(A)</code>
Comprimento do vetor	<code>length(A)</code>

OPERAÇÕES COM VETORES E MATRIZES

- **CRIAÇÃO DE MATRIZES**

- Matrizes são objetos 2D de tamanho $n \times m$
- No MATLAB, podemos criá-las de maneira similar aos vetores:

Command Window

```
>> A = [1 2 3; 4 5 6; 7 8 9]
```

A =

1	2	3
4	5	6
7	8	9

```
>> A = [1:3;4:6;7:9]
```

A =

1	2	3
4	5	6
7	8	9

OPERAÇÕES COM VETORES E MATRIZES

- **ALGUMAS OPERAÇÕES COM MATRIZES**

- Acessamos elementos de matrizes de maneira similar aos vetores, porém agora endereçando primeiro a linha e depois a coluna

```
>> A = [1 2 3; 4 5 6; 7 8 9];  
>> A(2,3)
```

```
ans =
```

```
6
```

```
>> A = [1 2 3; 4 5 6; 7 8 9];  
>> A(2:3,1:3)
```

```
ans =
```

```
4     5     6  
7     8     9
```

OPERAÇÕES COM VETORES E MATRIZES

- **ALGUMAS OPERAÇÕES COM MATRIZES**

- Para as operações de soma e subtração entre duas matrizes, lembrar que elas precisam ter as mesmas dimensões para produzir um resultado definido
- Para multiplicação entre matrizes, as dimensões de colunas de M e de linhas de N devem ser iguais:

$$M_{a,b} * N_{c,d}$$

- Alternativamente, temos a multiplicação ponto-a-ponto no MATLAB, que é feita com “.*”

OPERAÇÕES COM VETORES E MATRIZES

- OPERAÇÕES COM MATRIZES: VETORIAL VS PONTO-A-PONTO

```
>> A = [1 2 3; 4 5 6; 7 8 9]
```

A =

1	2	3
4	5	6
7	8	9

```
>> B = [2 4 6; 1 3 5; 1 2 3]
```

B =

2	4	6
1	3	5
1	2	3

OPERAÇÕES COM VETORES E MATRIZES

- OPERAÇÕES COM MATRIZES: VETORIAL VS PONTO-A-PONTO

```
>> A * B
```

```
ans =
```

7	16	25
19	43	67
31	70	109

```
>> A .* B
```

```
ans =
```

2	8	18
4	15	30
7	16	27

OPERAÇÕES COM VETORES E MATRIZES

- ALÉM DAS OPERAÇÕES JÁ MENCIONADAS PARA VETORES

Operação	Código
Inversão	$\text{inv}(A)$ ou A^{-1}
Divisão pela Esquerda Equivale a: $\text{inv}(A) * B$	$A \setminus B$
Divisão pela Direita Equivale a: $B * \text{inv}(A)$	B / A
Dimensões da matriz	$\text{size}(A)$

OPERAÇÕES COM VETORES E MATRIZES

- CRIAÇÃO DE VETORES E MATRIZES COM FUNÇÕES

- Algumas funções que criam matrizes úteis são

```
>> A = ones(2, 5)
```

A =

1	1	1	1	1
1	1	1	1	1

```
>> A = zeros(3, 4)
```

A =

0	0	0	0
0	0	0	0
0	0	0	0

Obs: chamando estas funções com uma das dimensões igual a 1, podemos criar vetores

OPERAÇÕES COM VETORES E MATRIZES

- CRIAÇÃO DE VETORES E MATRIZES COM FUNÇÕES

- Algumas funções que criam matrizes úteis são

```
>> A = eye(4, 4)
```

```
A =
```

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1



```
128 }
129 }
130 }
131 }
132 em.mail{
133     background: url(../img/mailico.png) no-repeat center;
134     display: inline-block;
135     width: 12px;
136     height: 14px;
137     float: left;
138     margin: 2px 7px 0 0;
139 }
140 em.phone{
141     background: url(../img/phoneico.png) no-repeat center;
142     display: inline-block;
143     width: 20px;
144     height: 18px;
145     float: left;
146     margin: 2px 7px 0 0;
147 }
```

POR HOJE É ISSO!

RELEMBRANDO A AGENDA PARA AMANHÃ...

AGENDA

- APRESENTAÇÃO DO PROGRAMA
- CONCEITOS BÁSICOS DE PROGRAMAÇÃO
- CRIAÇÃO DE SCRIPTS E FUNÇÕES CUSTOMIZADAS
- OPERAÇÕES COM VETORES E MATRIZES
- GERAÇÃO E FILTRAGEM DE SINAIS
- CRIAÇÃO DE GRÁFICOS
- TRABALHAR COM DADOS .TXT E .CSV
- EXPORTAR PROJETOS COMO RELATÓRIOS
- EXEMPLOS DE PROJETOS EM MATLAB NO ESCOPO DA GRADUAÇÃO

AULA 01

AULA 02



```
128 }
129 }
130 }
131 }
132 em.mail{
133     background: url(../img/mailico.png) no-repeat center;
134     display: inline-block;
135     width: 12px;
136     height: 14px;
137     float: left;
138     margin: 2px 7px 0 0;
139 }
140 em.phone{
141     background: url(../img/phoneico.png) no-repeat center;
142     display: inline-block;
143     width: 20px;
144     height: 18px;
145     float: left;
146     margin: 2px 7px 0 0;
147 }
```

OBRIGADO!
DÚVIDAS?