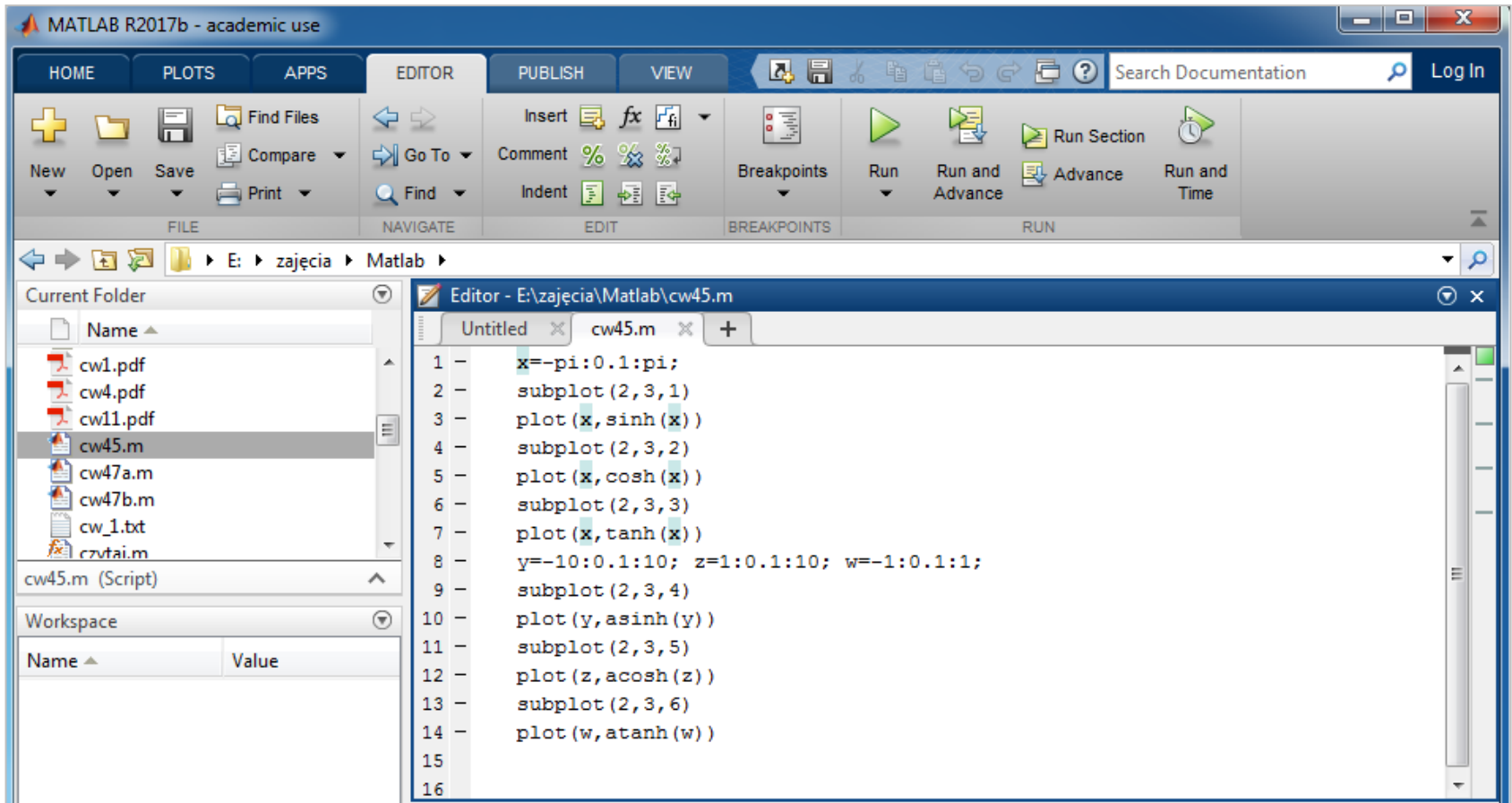


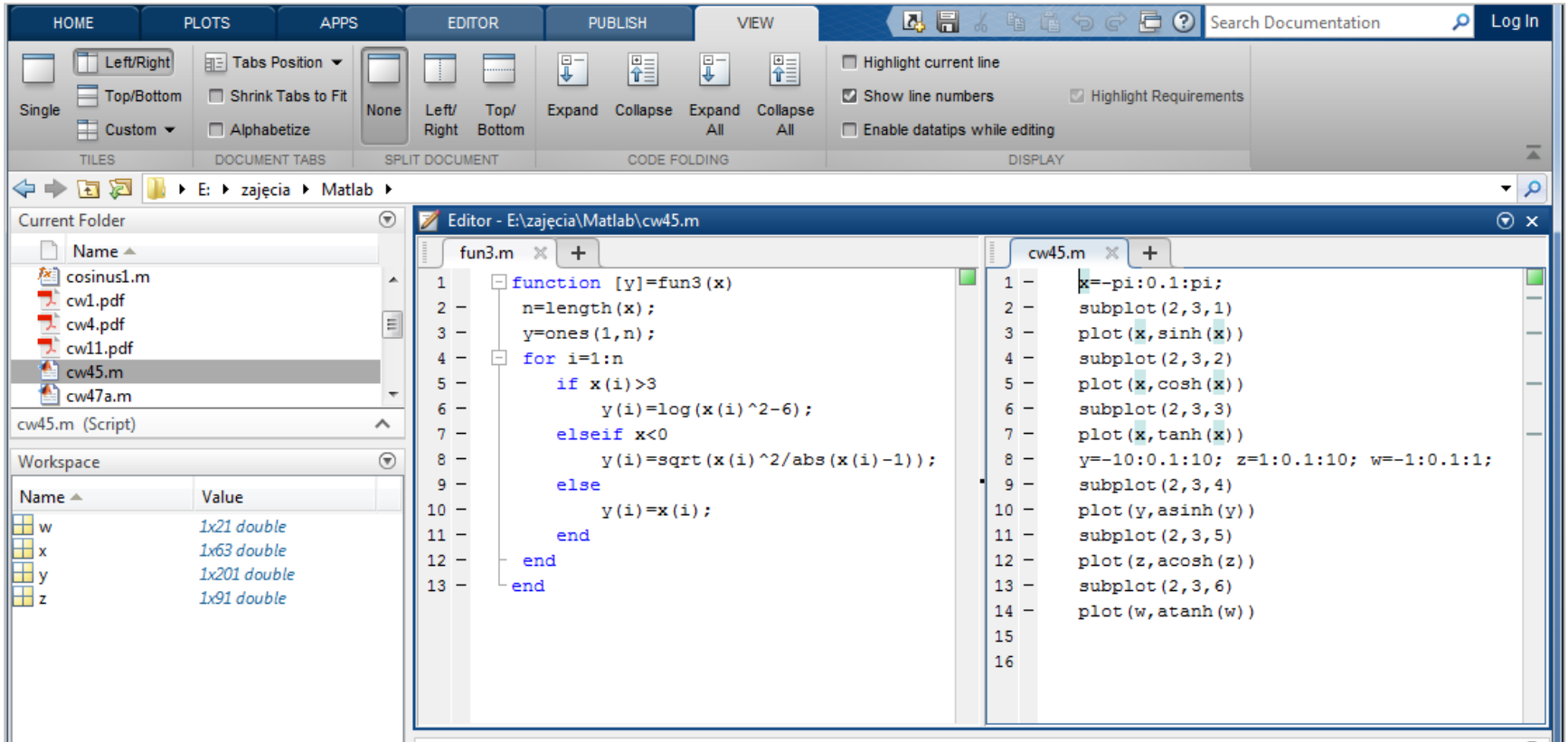
# *Skrypty i funkcje*

- ❖ Zapisywane są w m-plikach
- ❖ Wywoływane są przez nazwę m-pliku, w którym są zapisane (bez rozszerzenia)
- ❖ M-pliki mogą zawierać komentarze poprzedzone znakiem %
- ❖ **Skrypty**
  - służą do zapisu szeregu poleceń
  - działają tylko na zmiennych z przestrzeni roboczej
  - utworzone przez nie nowe zmienne zapisywane są w przestrzeni roboczej i pozostają w niej po wykonaniu skryptu
  - praktycznie można je traktować (i często tak się je tworzy) jako zapis fragmentu okna historii poleceń



The image shows the MATLAB R2017b - academic use interface. The window title is "MATLAB R2017b - academic use". The interface includes a ribbon with tabs for HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW. The EDITOR tab is active, showing a toolbar with icons for New, Open, Save, Find Files, Compare, Print, Go To, Find, Insert, Comment, Indent, Breakpoints, Run, Run and Advance, Run Section, and Run and Time. The current folder is "E:\zajęcia\Matlab". The workspace is empty. The editor window shows the following code:

```
1 - x=-pi:0.1:pi;  
2 - subplot(2,3,1)  
3 - plot(x,sinh(x))  
4 - subplot(2,3,2)  
5 - plot(x,cosh(x))  
6 - subplot(2,3,3)  
7 - plot(x,tanh(x))  
8 - y=-10:0.1:10; z=1:0.1:10; w=-1:0.1:1;  
9 - subplot(2,3,4)  
10 - plot(y,asinh(y))  
11 - subplot(2,3,5)  
12 - plot(z,acosh(z))  
13 - subplot(2,3,6)  
14 - plot(w,atanh(w))  
15  
16
```



The screenshot displays the MATLAB Editor environment. The top ribbon includes tabs for HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW. The EDITOR tab is active, showing various editing and display options.

The left sidebar shows the Current Folder (E:\zajęcia\Matlab) and Workspace. The Workspace contains variables w (1x21 double), x (1x63 double), y (1x201 double), and z (1x91 double).

The main editor window shows two files: fun3.m and cw45.m.

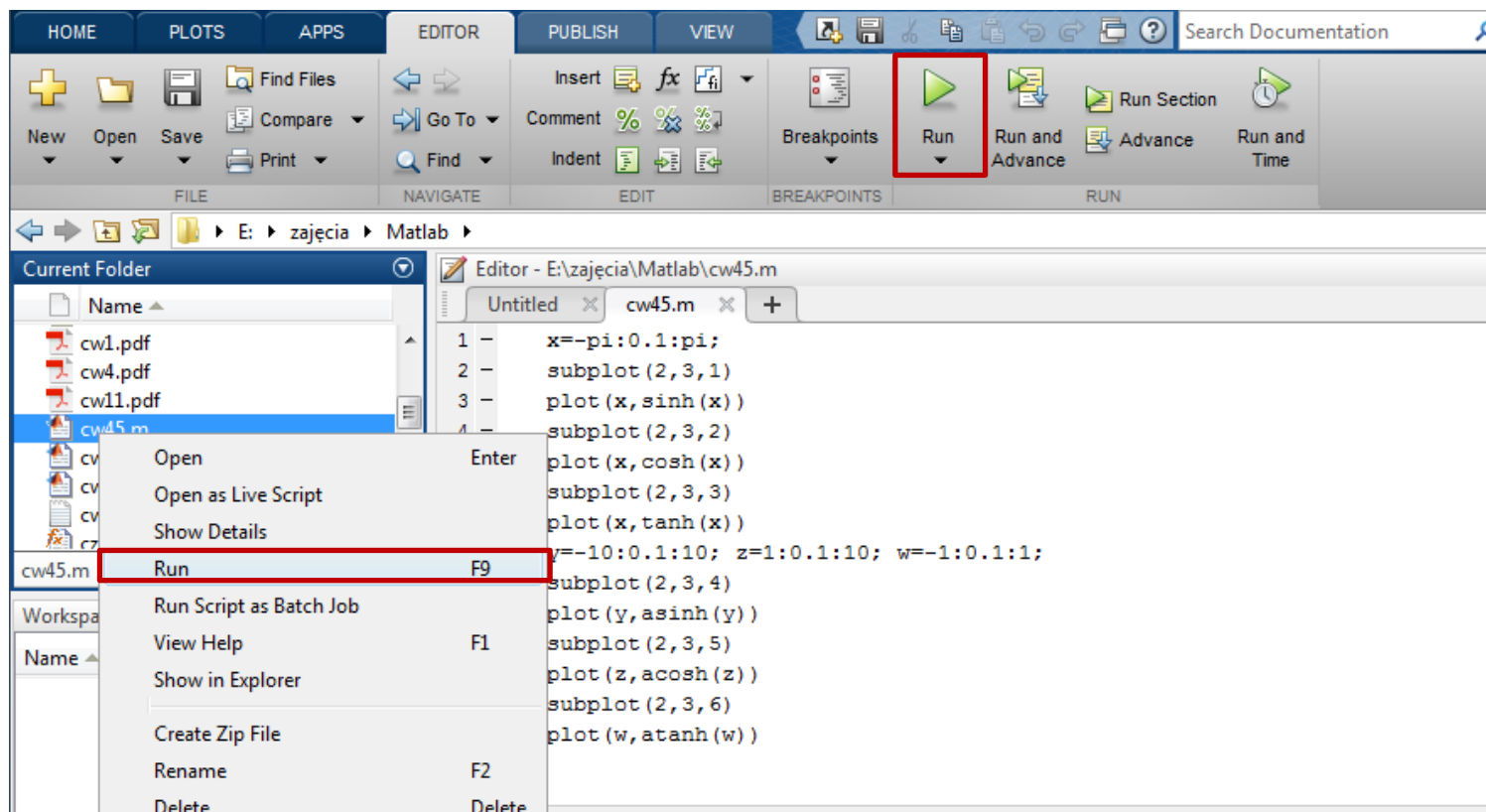
```

1 function [y]=fun3(x)
2     n=length(x);
3     y=ones(1,n);
4     for i=1:n
5         if x(i)>3
6             y(i)=log(x(i)^2-6);
7         elseif x<0
8             y(i)=sqrt(x(i)^2/abs(x(i)-1));
9         else
10            y(i)=x(i);
11        end
12    end
13 end
  
```

```

1 x=-pi:0.1:pi;
2 subplot(2,3,1)
3 plot(x,sinh(x))
4 subplot(2,3,2)
5 plot(x,cosh(x))
6 subplot(2,3,3)
7 plot(x,tanh(x))
8 y=-10:0.1:10; z=1:0.1:10; w=-1:0.1:1;
9 subplot(2,3,4)
10 plot(y,asinh(y))
11 subplot(2,3,5)
12 plot(z,acosh(z))
13 subplot(2,3,6)
14 plot(w,atanh(w))
15
16
  
```

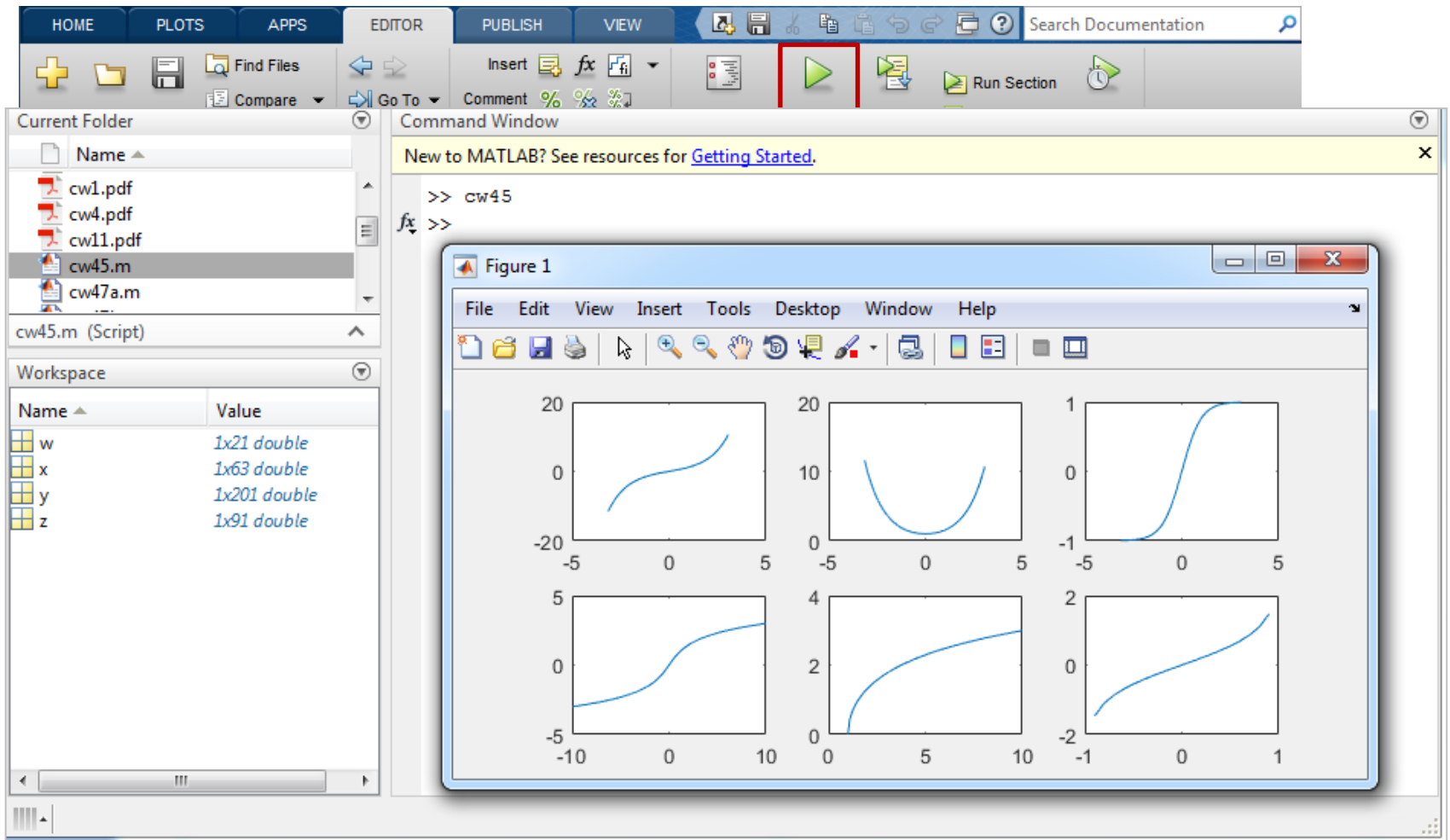
# Skrypty -uruchamianie



The screenshot displays the MATLAB software interface. The top toolbar features several tabs: HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW. The EDITOR tab is active, showing a menu with options like New, Open, Save, and Print. The Run button, represented by a green play icon, is highlighted with a red box. Below the toolbar, the current folder is 'E:\zajęcia\Matlab'. A context menu is open over the file 'cw45.m', with the 'Run' option highlighted in blue and a red box. The 'Run' option is associated with the keyboard shortcut F9. The editor window shows the following MATLAB code:

```
1 x=-pi:0.1:pi;
2 subplot(2,3,1)
3 plot(x,sinh(x))
4 subplot(2,3,2)
5 plot(x,cosh(x))
6 subplot(2,3,3)
7 plot(x,tanh(x))
8 y=-10:0.1:10; z=1:0.1:10; w=-1:0.1:1;
9 subplot(2,3,4)
10 plot(y,asinh(y))
11 subplot(2,3,5)
12 plot(z,acosh(z))
13 subplot(2,3,6)
14 plot(w,atanh(w))
```

# Skrypty -uruchamianie



The image displays the MATLAB software interface. The top menu bar includes HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW. The toolbar contains various icons, with the Run button (a green play icon) highlighted by a red rectangle. Below the toolbar, the 'Current Folder' pane shows a list of files: cw1.pdf, cw4.pdf, cw11.pdf, cw45.m (selected), and cw47a.m. The 'Workspace' pane shows variables w (1x21 double), x (1x63 double), y (1x201 double), and z (1x91 double). The 'Command Window' displays the command `>> cw45` and the output `fx >>`. A yellow banner at the top of the Command Window reads 'New to MATLAB? See resources for [Getting Started](#).' The 'Figure 1' window contains six subplots arranged in a 2x3 grid. The top row shows: a plot of a curve on a coordinate system with x-axis from -5 to 5 and y-axis from -20 to 20; a plot of a parabolic curve with x-axis from -5 to 5 and y-axis from 0 to 20; and a plot of a sigmoid function with x-axis from -5 to 5 and y-axis from -1 to 1. The bottom row shows: a plot of a curve on a coordinate system with x-axis from -10 to 10 and y-axis from -5 to 5; a plot of a curve on a coordinate system with x-axis from 0 to 10 and y-axis from 0 to 4; and a plot of a curve on a coordinate system with x-axis from -1 to 1 and y-axis from -2 to 2.

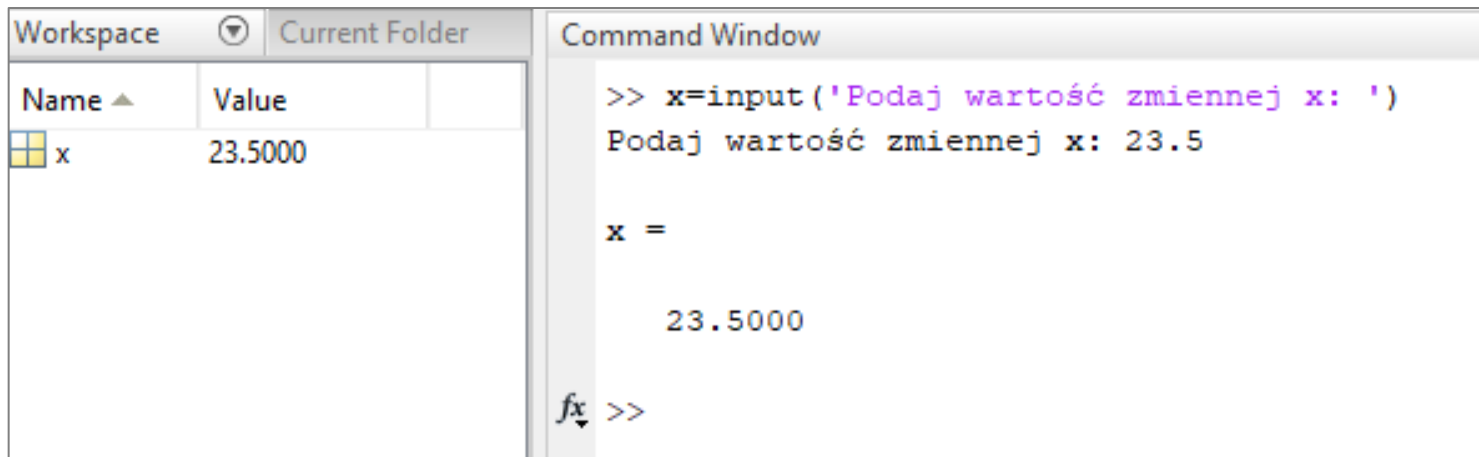
# Obsługa wejścia/wyjścia skryptu

## ❖ Funkcje obsługi wejścia skryptu:

- ***zm=input('tekst')*** – wyświetla łańcuch ***tekst***, oczekuje na wpisanie przez użytkownika ***danej liczbowej*** i przypisuje ją zmiennej ***zm***
- ***zm=input('tekst','s')*** – wyświetla łańcuch ***tekst***, oczekuje na wpisanie przez użytkownika ***łańcucha znakowego*** i przypisuje ją zmiennej ***zm***
- ***wybor=menu('Opis', 'Przycisk1', 'Przycisk2',...)***
- ***pause*** – zatrzymuje wykonywanie skryptu do momentu naciśnięcia przez użytkownika dowolnego klawisza
- ***pause(n)*** – zatrzymuje wykonywanie skryptu przez ***n*** sekund

# Obsługa wejścia/wyjścia skryptu

```
>> x=input('Podaj wartość zmiennej x: ')\nPodaj wartość zmiennej x:
```



The image shows a screenshot of the MATLAB interface. On the left, the 'Workspace' window displays a table with two columns: 'Name' and 'Value'. A variable 'x' is listed with a value of 23.5000. On the right, the 'Command Window' shows the execution of the input command. The prompt 'Podaj wartość zmiennej x: ' is followed by the user input '23.5'. The output shows 'x = 23.5000'.

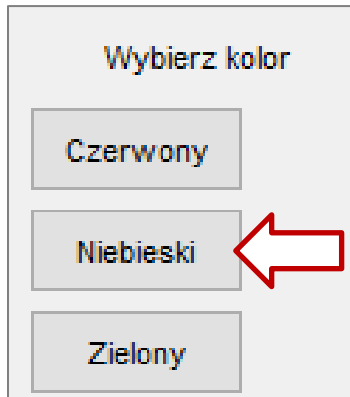
Name	Value
x	23.5000

```
>> x=input('Podaj wartość zmiennej x: ')\nPodaj wartość zmiennej x: 23.5\n\nx =\n\n    23.5000\n\nfx >>
```



# Obsługa wejścia/wyjścia skryptu

```
>> kolor=menu('Wybierz kolor', 'Czerwony', 'Niebieski', 'Zielony')
```



```
>> kolor=menu('Wybierz kolor', 'Czerwony', 'Niebieski', 'Zielony')
```

```
kolor =
```

## ❖ Funkcje obsługi wyjścia skryptu:

- ***disp(x)*** – wyświetla na ekranie wartość zmiennej *x*
- ***disp('s')*** – wyświetla na ekranie wartość zmiennej łańcuch znakowy *s*
- ***fprintf(format, zm1, zm2, ...)*** – wyświetla na ekranie ciąg znaków zawierający tekst oraz sformatowane wartości zmiennych
- ***str=sprintf(format, zm1, zm2, ...)*** – zwraca ciąg znaków zawierający tekst oraz sformatowane wartości zmiennych

***format*** – ciąg znaków zawierający tekst oraz operatory formatowania,

np. %d, %i – liczby całkowite ze znakiem

%f – liczby rzeczywiste

%s – łańcuch znakowy

%c – znak

# łańcuchy znakowe

- ❖ Przykładowe funkcje przetwarzające łańcuchy znakowe:
  - ***strcat(s1, s2, s3, ...)*** – łączy w poziomie łańcuchy znakowe
  - ***strvcat(s1, s2, s3)*** – łączy w pionie łańcuchy znakowe
  - ***upper(s)*** – zmienia wszystkie litery na duże
  - ***lower(s)*** – zmienia wszystkie litery na małe
  - ***deblank(s)*** – usuwa spacje z końca łańcucha
  - ***strcmp(s1,s2)*** – porównuje dwa łańcuchy znakowe, jeżeli są identyczne zwraca 1, jeśli nie 0
- ❖ Przykładowe funkcje konwertujące łańcuchy znakowe:
  - ***int2str(n)*** – konwertuje liczbę całkowitą na łańcuch znakowy
  - ***num2str(x)*** – konwertuje wyrażenie Matlaba na łańcuch znakowy
  - ***str2double(s)*** – konwertuje łańcuch znakowy na liczbę

# łańcuchy znakowe

```
Command Window
>> x=12.34,y=12.34e-5,s='program',s1='56.78'

x =

    12.3400

y =

    1.2340e-04

s =

    program

s1 =

    56.78

fx >>
```

Workspace

Name ▲	
abc	s
abc	s1
abc	s2
abc	s3
abc	s4
	x
	x1
	x2
	y

```
Command Window
>> x1=str2double(s)

x1 =

    NaN

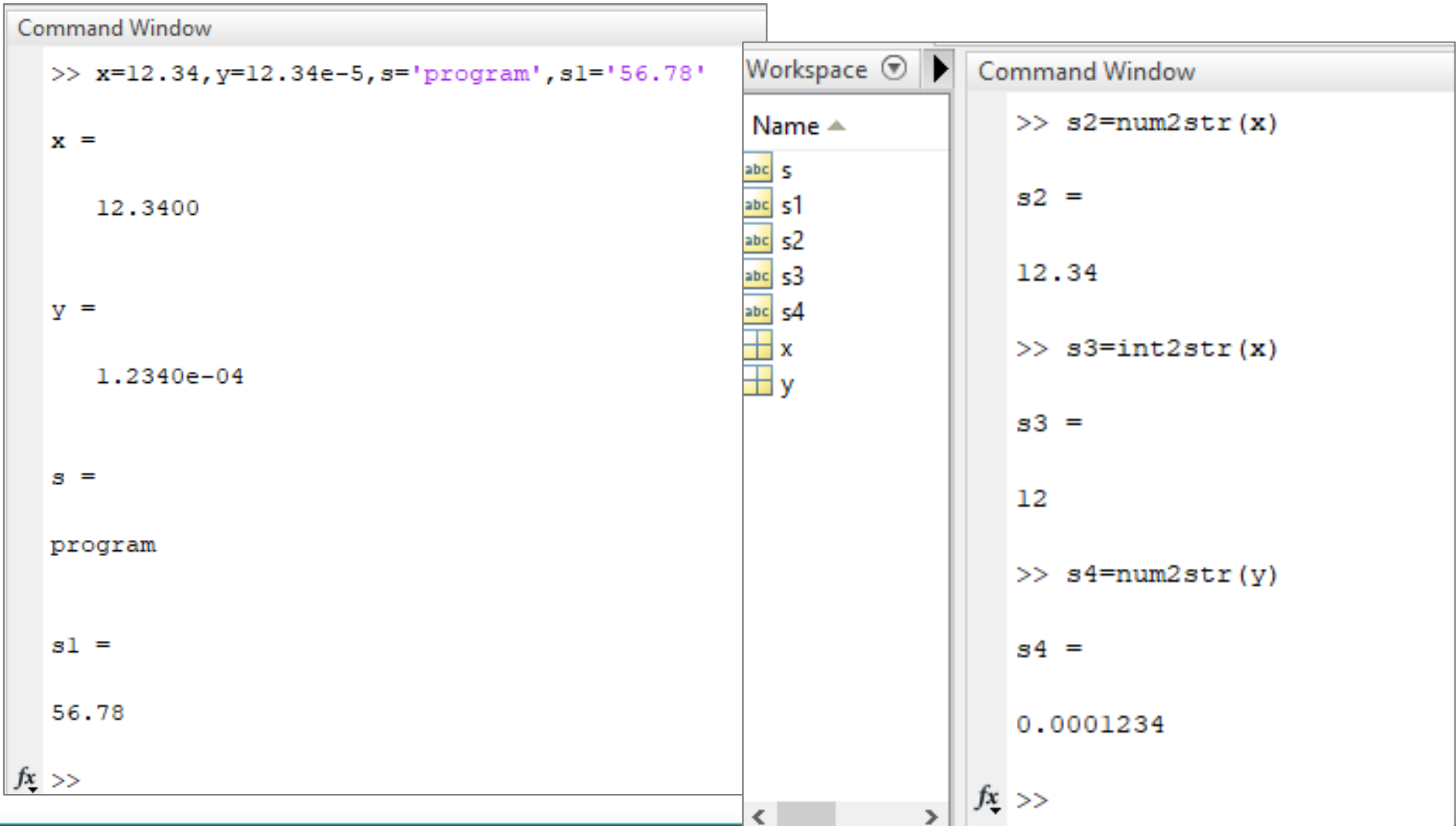
>> x2=str2double(s1)

x2 =

    56.7800

fx >>
```

# łańcuchy znakowe



The image shows two screenshots of the MATLAB Command Window and Workspace. The left screenshot shows the initial assignment of variables x, y, s, and s1. The right screenshot shows the conversion of x and y to strings using num2str and int2str functions. The Workspace window in the middle shows the current state of the workspace with variables s, s1, s2, s3, s4, x, and y.

```
>> x=12.34,y=12.34e-5,s='program',s1='56.78'
```

x =  
12.3400

y =  
1.2340e-04

s =  
program

s1 =  
56.78

```
>> s2=num2str(x)
```

s2 =  
12.34

```
>> s3=int2str(x)
```

s3 =  
12

```
>> s4=num2str(y)
```

s4 =  
0.0001234

Workspace

Name
s
s1
s2
s3
s4
x
y

# łańcuchy znakowe

```
>> s1='tekst 1', s2='test2'  
  
s1 =  
tekst 1  
  
s2 =  
test2  
  
>> s3=strcat(s1,s2)  
  
s3 =  
tekst 1test2  
  
>> s4=strvcat(s1,s2)  
  
s4 =  
tekst 1  
test2  
  
>> disp(strcat('Zmienna s1 ma wartość ',s1,', a zmienna s2 ma wartość ',s2))  
Zmienna s1 ma wartośćtekst 1. a zmienna s2 ma wartośćtest2
```

# łańcuchy znakowe

```
>> a=1, b=12.5
```

```
a =
```

```
1
```

```
b =
```

```
12.5000
```

```
>> disp(strcat('Zmienna a ma wartość '→a,', a zmienna b ma wartość '→b))
```

```
Zmienna a ma wartość[← a zmienna b ma wartość[←
```

```
>> disp(strcat('Zmienna a ma wartość:'→num2str(a),', a zmienna b ma wartość:'→num2str(b)))
```

```
Zmienna a ma wartość:1, a zmienna b ma wartość:12.5←
```

```
>>
```

# Obsługa wejścia/wyjścia skryptu

```
>> X=12, Y=12.3456, s='program'
```

```
X =
```

```
12
```

```
Y =
```

```
12.3456
```

```
s =
```

```
program
```

```
fx >>
```

```
>> disp(X)
```

```
12
```

```
>> disp(s)
```

```
program
```

```
>> disp('tekst')
```

```
tekst
```

```
>> fprintf('Wartość zmiennej X wynosi %d, zmiennej Y wynosi %.3f, a zmiennej s wynosi %s',X,Y,s)
```

```
Wartość zmiennej X wynosi 12, zmiennej Y wynosi 12.346, a zmiennej s wynosi program>>
```

```
>> fprintf('Wartość zmiennej X wynosi %d, zmiennej Y wynosi %.3f, a zmiennej s wynosi %s. \n',X,Y,s)
```

```
Wartość zmiennej X wynosi 12, zmiennej Y wynosi 12.346, a zmiennej s wynosi program.
```

```
>> sprintf('Wartość zmiennej X wynosi %d, zmiennej Y wynosi %.3f, a zmiennej s wynosi %s.',X,Y,s)
```

```
ans =
```

```
Wartość zmiennej X wynosi 12, zmiennej Y wynosi 12.346, a zmiennej s wynosi program.
```

```
>> disp(ans)
```

```
Wartość zmiennej X wynosi 12, zmiennej Y wynosi 12.346, a zmiennej s wynosi program.
```

```
>>
```



## ❖ Funkcje

- działają poprzez argumenty wejściowe i wyjściowe, pobierane z (te pierwsze) i zwracane do (te drugie) obszaru roboczego
- argumenty wejściowe mogą być modyfikowane
- każda wywołana funkcja posiada własny obszar pamięci – obszar roboczy funkcji
- zmienne lokalne funkcji przechowywane są w jej obszarze roboczym i nie są dostępne dla innych funkcji czy z linii poleceń
- można też zdefiniować zmienne globalne, które będą dostępne z przestrzeni roboczej (konwencjonalnie takie zmienne mają nazwy pisane wierszalikami) – składnia: ***global ZMIENNE***
- nazwa pliku z funkcją powinna być taka sama jak nazwa funkcji

## Składnia funkcji

- ❖ Pierwsza linia m-pliku definiującej funkcję (linia definicji) ma ustaloną składnię:

***function [out1,...,outn]=nazwa\_funkcji(inp1,...,inpm)***

składnia, gdy jest tylko jeden argument wyjściowy:

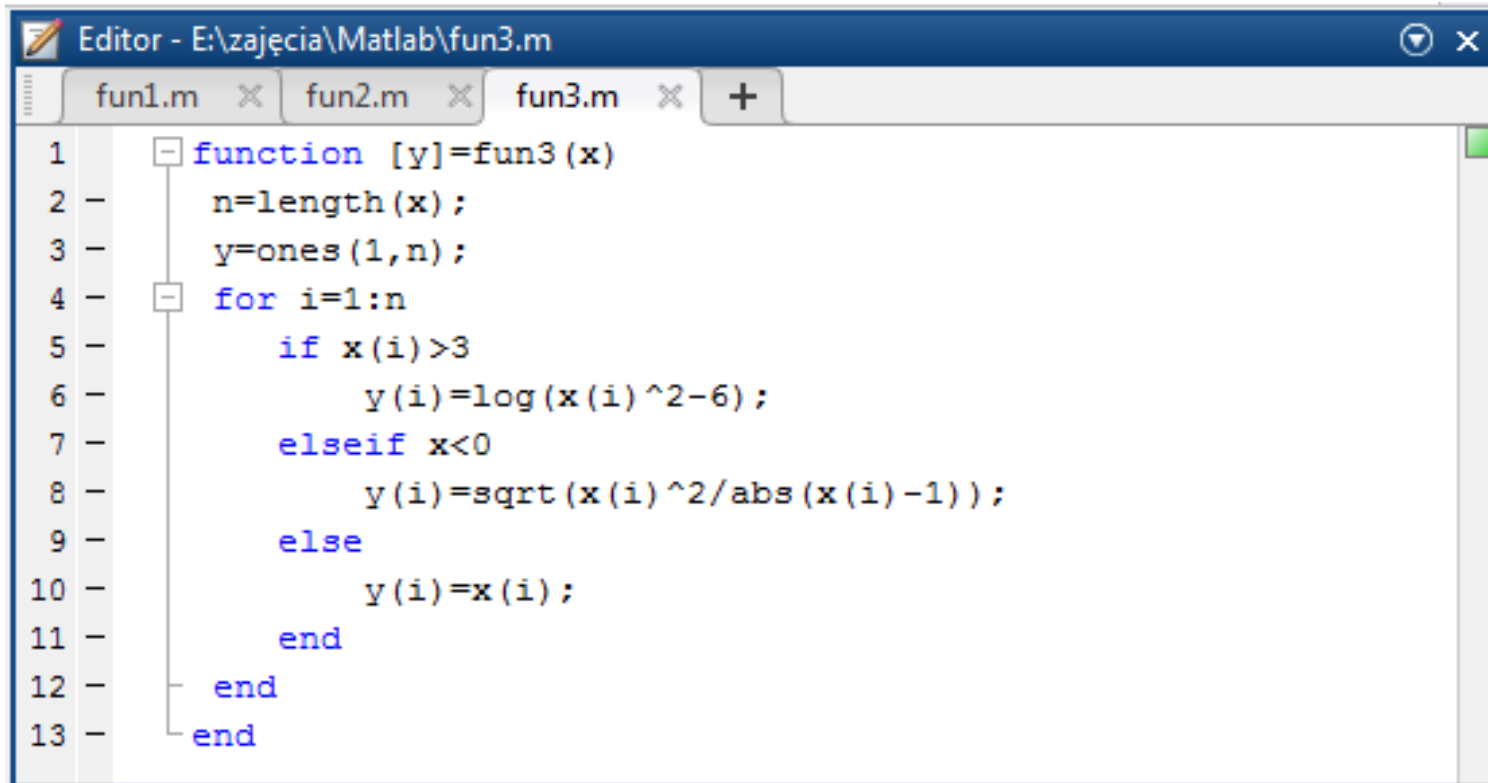
***function out=nazwa\_funkcji(inp1,...,inpm)***

może również nie być argumentu wyjściowego:

***function nazwa\_funkcji(inp1,...,inpm)***

- ❖ Kolejnym elementem są wiersze nagłówka (kilka linii komentarza), które wyświetlane są w oknie poleceń po poleceniu: ***help nazwa\_m-pliku\_funkcji***
- ❖ Pierwsza linia nagłówka ma szczególne znaczenie – jest przeszukiwana przez funkcję ***lookfor***
- ❖ Po nagłówku następuje kod obliczeniowy funkcji

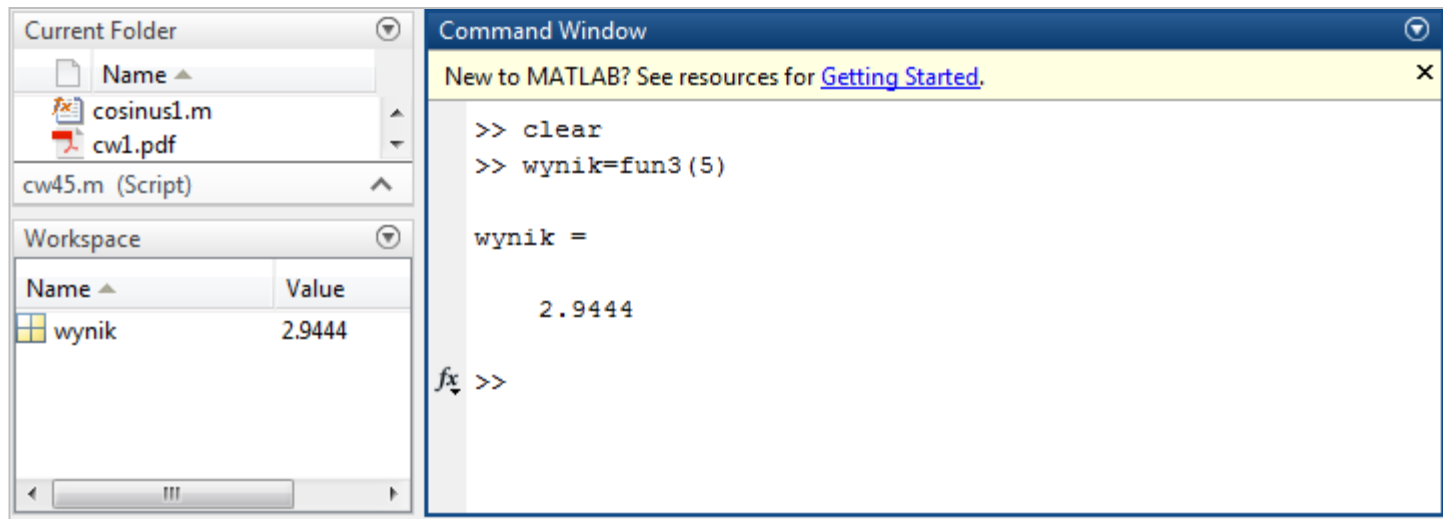
# Składnia funkcji



```
Editor - E:\zajęcia\Matlab\fun3.m
fun1.m x fun2.m x fun3.m x +
1 function [y]=fun3(x)
2     n=length(x);
3     y=ones(1,n);
4     for i=1:n
5         if x(i)>3
6             y(i)=log(x(i)^2-6);
7         elseif x<0
8             y(i)=sqrt(x(i)^2/abs(x(i)-1));
9         else
10            y(i)=x(i);
11        end
12    end
13 end
```

# Funkcje -uruchamianie

- ❖ Wywołanie (uruchomienie) funkcji polega na wpisaniu w Command Window nazwy funkcji z odpowiednimi argumentami i naciśnięci Enter
- ❖ Argumenty funkcji wpisujemy w nawiasach i oddzielamy przecinkami



The screenshot shows the MATLAB interface. On the left, the 'Current Folder' pane displays files: 'cosinus1.m', 'cw1.pdf', and 'cw45.m (Script)'. Below it, the 'Workspace' pane shows a table with one variable:

Name	Value
wynik	2.9444

On the right, the 'Command Window' shows the following commands and output:

```
New to MATLAB? See resources for Getting Started.
>> clear
>> wynik=fun3(5)

wynik =

    2.9444

fx >>
```

## Zmienna liczba „wejść” i „wyjść”

- ❖ Pomimo deklaracji liczby „wejść” i „wyjść” funkcji, można w wywołaniu funkcji użyć mniejszej liczby argumentów czy zmiennych, do których zwracamy wynik.
- ❖ W takim wypadku, aby uniknąć błędów wykonania funkcji, należy ten stan kontrolować funkcjami: *nargin* oraz *nargout* wewnątrz kodu funkcji.
- ❖ Przykład:

```
function rad = radiany(st,min,sek)
if nargin == 1
    rad = st*pi/180;
else
    rad = (st + min/60 + sek/360)*pi/180;
end
```

```
>> wynik1=radiany(30)

wynik1 =

    0.5236

>> wynik2=radiany(30,15,10)

wynik2 =

    0.5284

>> wynik3=radiany([30 90 180])

wynik3 =

    0.5236    1.5708    3.1416
```

## Zmienna liczba „wejść” i „wyjść”

- ❖ Można też w deklaracji funkcji przewidzieć zmienną liczbę „wejść” i „wyjść” za pomocą zmiennych: `varargin` oraz `varargout`. Zmienne te podajemy na końcu specyfikacji. Przyjmują typ wektora komórkowego o długości takiej, jak liczba nadmiarowych „wejść” lub „wyjść”.
- ❖ Przykład:  
$$\textit{function rad = radians(st,varargin)}$$
- ❖ W przypadku zadeklarowania **`varargin`** lub **`varargout`** funkcje **`nargin`** oraz **`nargout`** przyjmują wartość ujemną.
- ❖ W podanym przykładzie funkcja `nargin` zwróci wartość `-2`.

## Zmienna liczba „wejść” i „wyjść”

- ❖ Kontrolę zakresu liczby „wejść” i „wyjść” można realizować za pomocą funkcji:

*narginchk(min,max)* oraz *nargoutchk(min,max)*.

- ❖ Funkcja *narginchk* przy przekroczeniu zakresu liczby wejść wyświetla komunikaty:

*'Not enough input arguments.'*

lub

*'Too many input arguments.'*

- ❖ Funkcja *nargoutchk* przy przekroczeniu zakresu liczby wyjść wyświetla komunikaty:

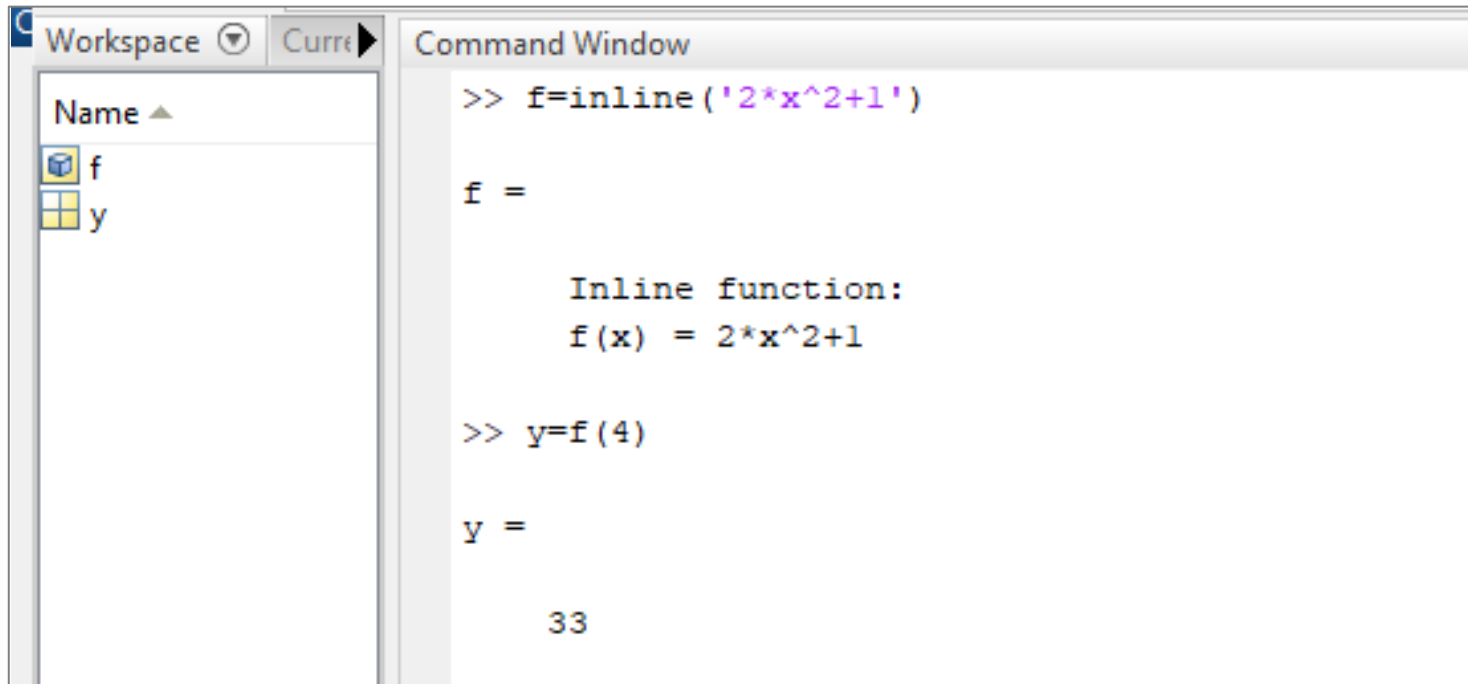
*'Not enough output arguments.'*

lub

*'Too many output arguments.'*

# Funkcja *inline*

- ❖ ***funkcja = inline('wyrażenie')*** – zdefiniowanie funkcji w wierszu
  - ***funkcja*** – nazwa zmiennej
  - ***wyrażenie*** – wzór funkcji w postaci łańcucha znakowego



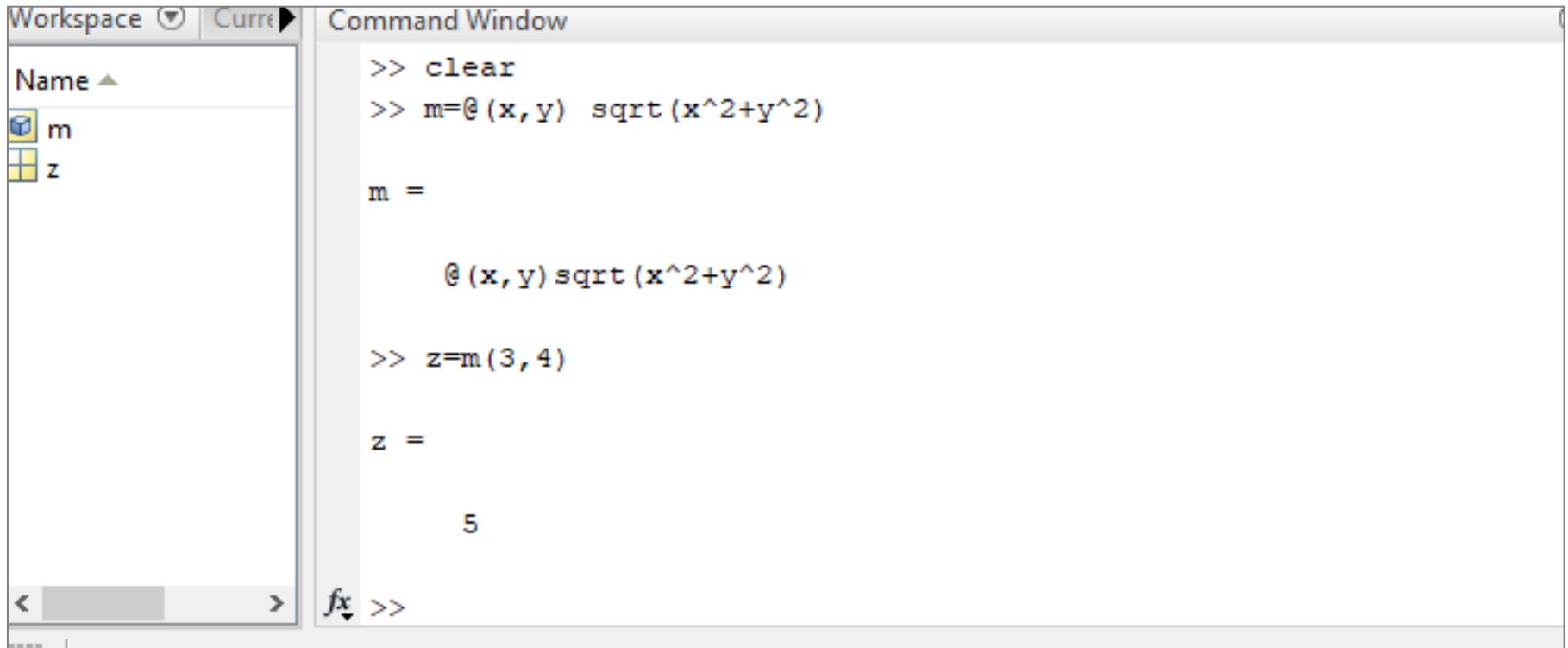
The screenshot shows the MATLAB Command Window interface. On the left, the 'Workspace' pane lists variables 'f' and 'y'. The 'Command Window' on the right shows the following interaction:

```
>> f=inline('2*x^2+1')  
  
f =  
  
    Inline function:  
    f(x) = 2*x^2+1  
  
>> y=f(4)  
  
y =  
  
    33
```



# Funkcja anonimowe

❖ *funkcja = @(argumenty funkcji) kod funkcji*



The screenshot shows the MATLAB Command Window interface. On the left, the 'Workspace' pane lists variables 'm' and 'z'. The 'Command Window' on the right contains the following code and output:

```
>> clear
>> m=@(x,y) sqrt(x^2+y^2)

m =

    @(x,y) sqrt(x^2+y^2)

>> z=m(3,4)

z =

     5

fx >>
```