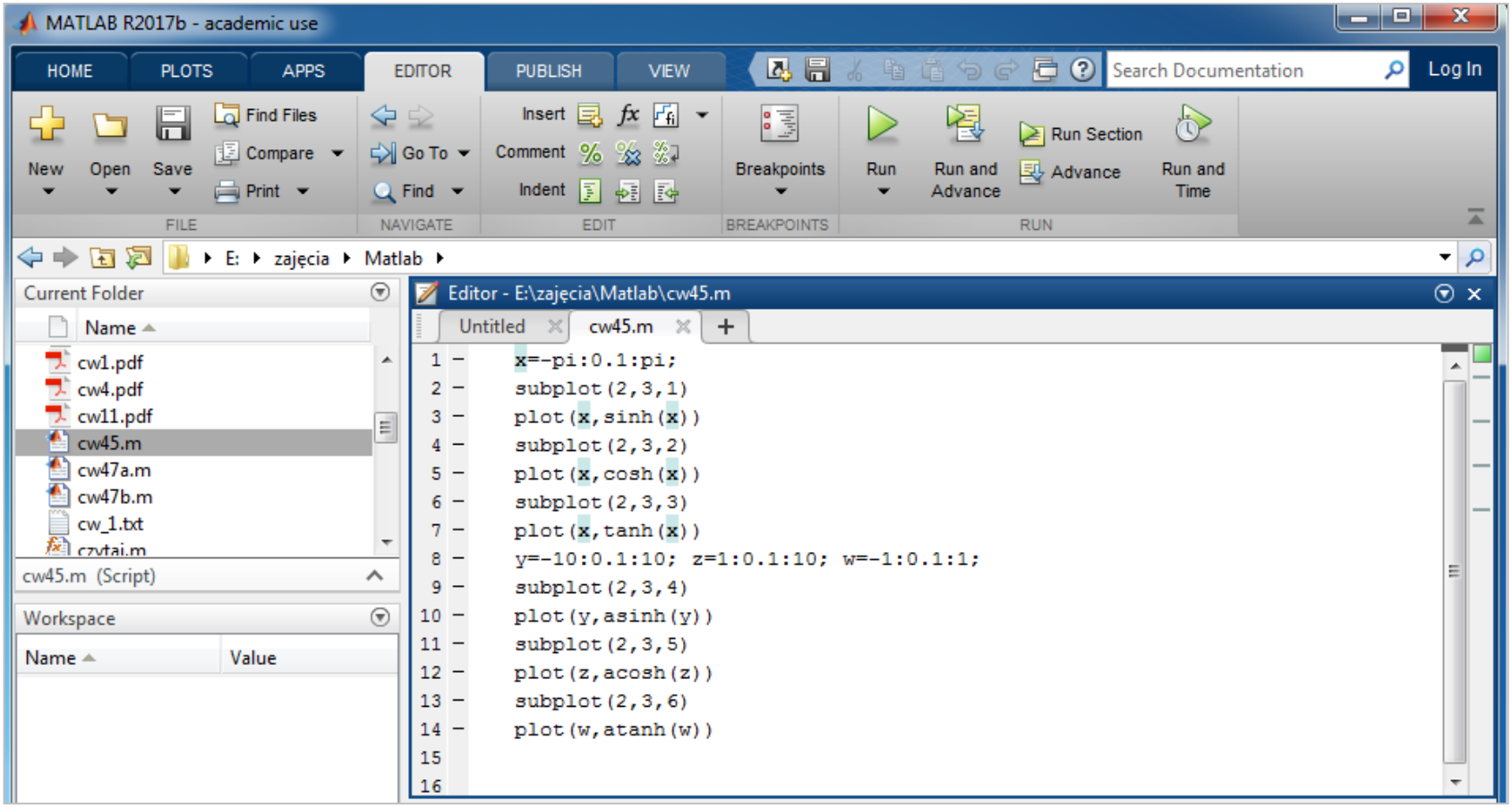
AGH

Skrypty i funkcje

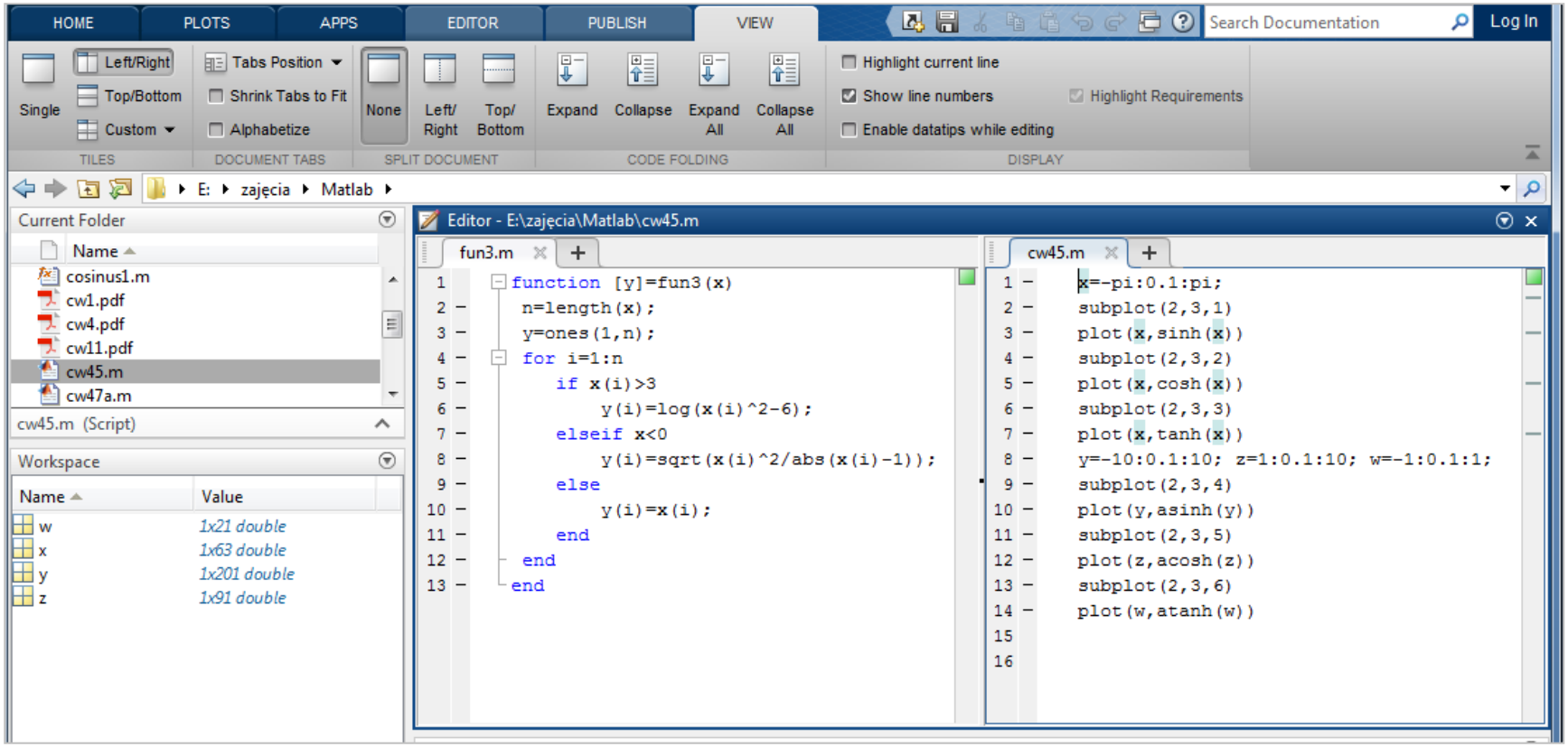
Skrypty i funkcje

- 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

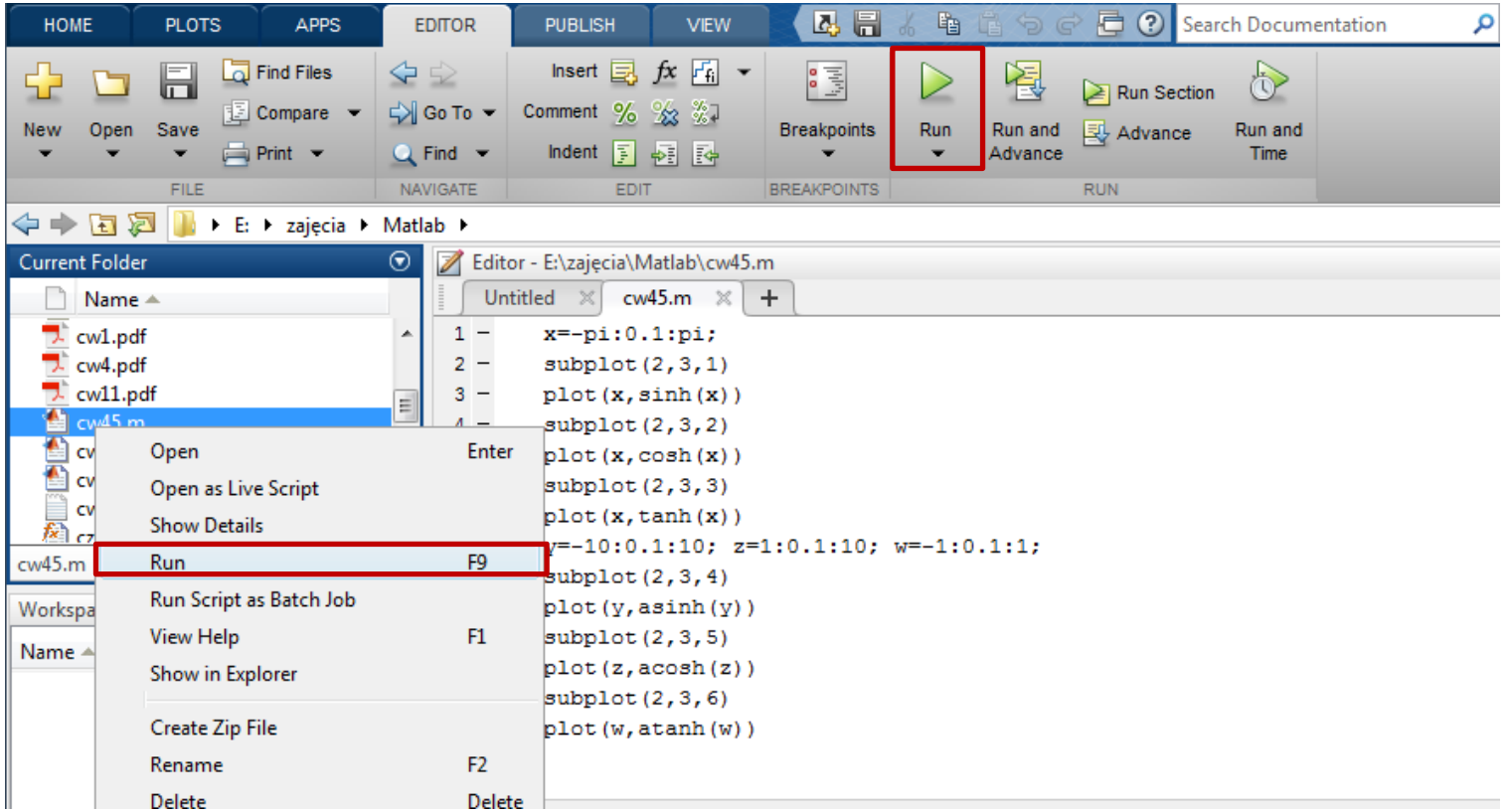
Skrypty


MATLAB R2017b - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Log In
New
Open
Save
Find Files
Compare
Print
Go To
Find
Insert
Comment
Indent
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
NAVIGATE
EDIT
BREAKPOINTS
RUN
E: ▸ zajęcia ▸ Matlab ▸
Current Folder
Name
cw1.pdf
cw4.pdf
cw11.pdf
cw45.m
cw47a.m
cw47b.m
cw_1.txt
czytaj.m
cw45.m (Script)
Workspace
Name
Value
Editor - E:\zajęcia\Matlab\cw45.m
Untitled
cw45.m
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

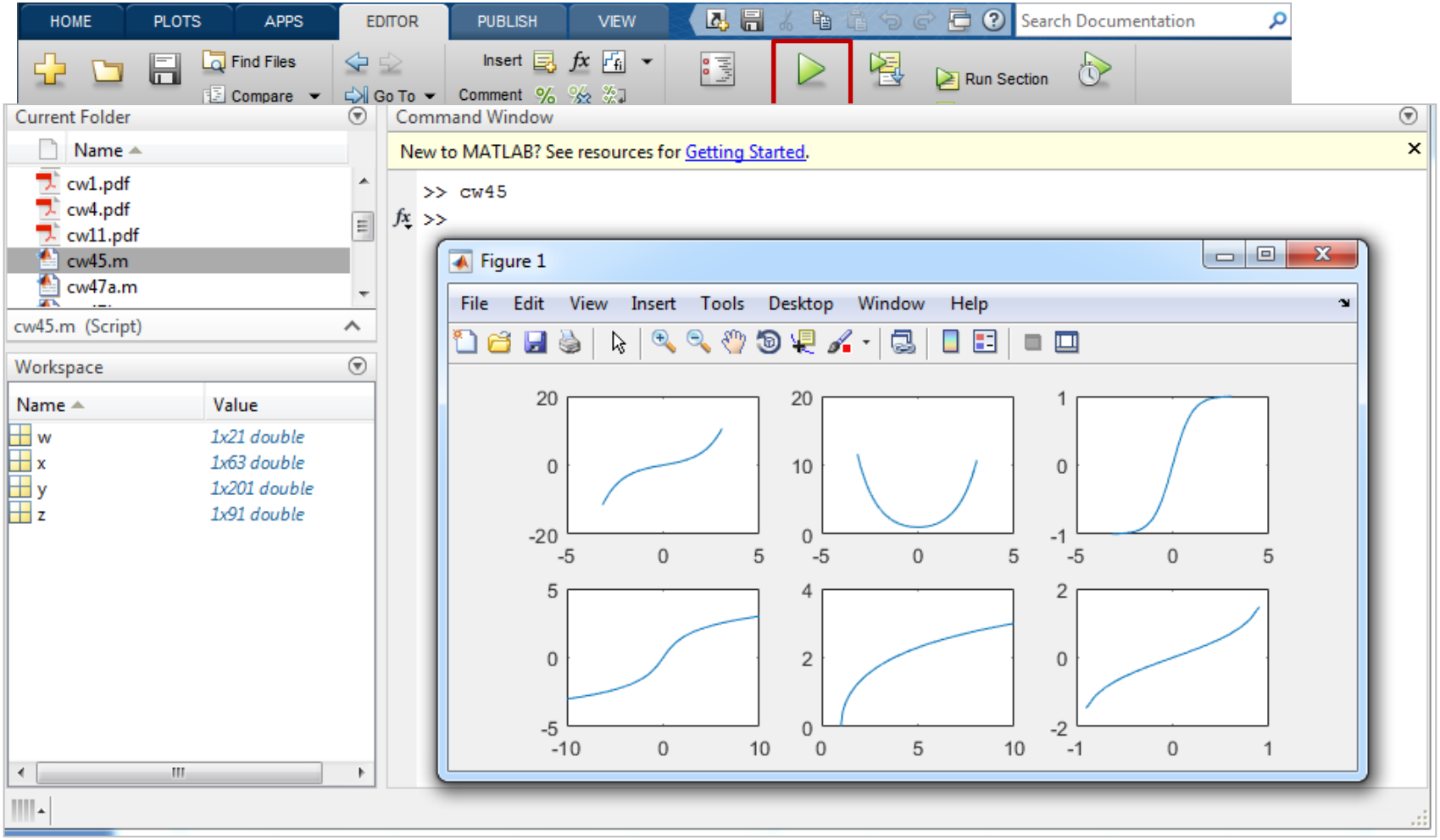
Edytor


HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Log In
Single
Left/Right
Top/Bottom
Custom
Tabs Position
Shrink Tabs to Fit
Alphabetize
None
Left/ Right
Top/ Bottom
Expand
Collapse
Expand All
Collapse All
Highlight current line
Show line numbers
Highlight Requirements
Enable datatips while editing
TILES
DOCUMENT TABS
SPLIT DOCUMENT
CODE FOLDING
DISPLAY
E: ▸ zajęcia ▸ Matlab ▸
Current Folder
Name
cosinus1.m
cw1.pdf
cw4.pdf
cw11.pdf
cw45.m
cw47a.m
cw45.m (Script)
Workspace
Name
Value
w 1x21 double
x 1x63 double
y 1x201 double
z 1x91 double
Editor - E:\zajęcia\Matlab\cw45.m
fun3.m
function [y]=fun3(x)
n=length(x);
y=ones(1,n);
for i=1:n
if x(i)>3
y(i)=log(x(i)^2-6);
elseif x<0
y(i)=sqrt(x(i)^2/abs(x(i)-1));
else
y(i)=x(i);
end
end
end
cw45.m
x=-pi:0.1:pi;
subplot(2,3,1)
plot(x,sinh(x))
subplot(2,3,2)
plot(x,cosh(x))
subplot(2,3,3)
plot(x,tanh(x))
y=-10:0.1:10; z=1:0.1:10; w=-1:0.1:1;
subplot(2,3,4)
plot(y,asinh(y))
subplot(2,3,5)
plot(z,acosh(z))
subplot(2,3,6)
plot(w,atanh(w))

Skrypty -uruchamianie



Skrypty -uruchamianie

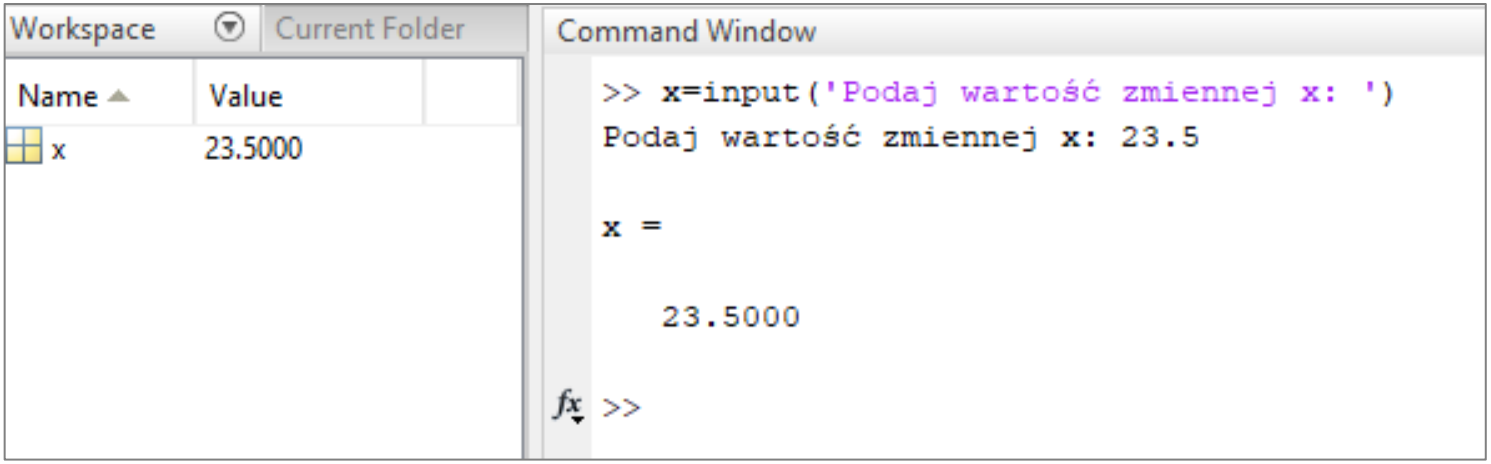


Obsługa wejścia/wyjścia skryptu

- 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

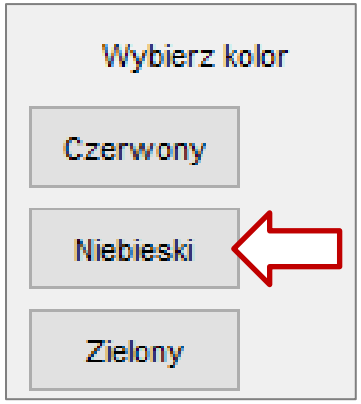
Obsługa wejścia/wyjścia skryptu

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



Obsługa wejścia/wyjścia skryptu

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




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

kolor =

     2
```

Obsługa wejścia/wyjścia skryptu

- 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

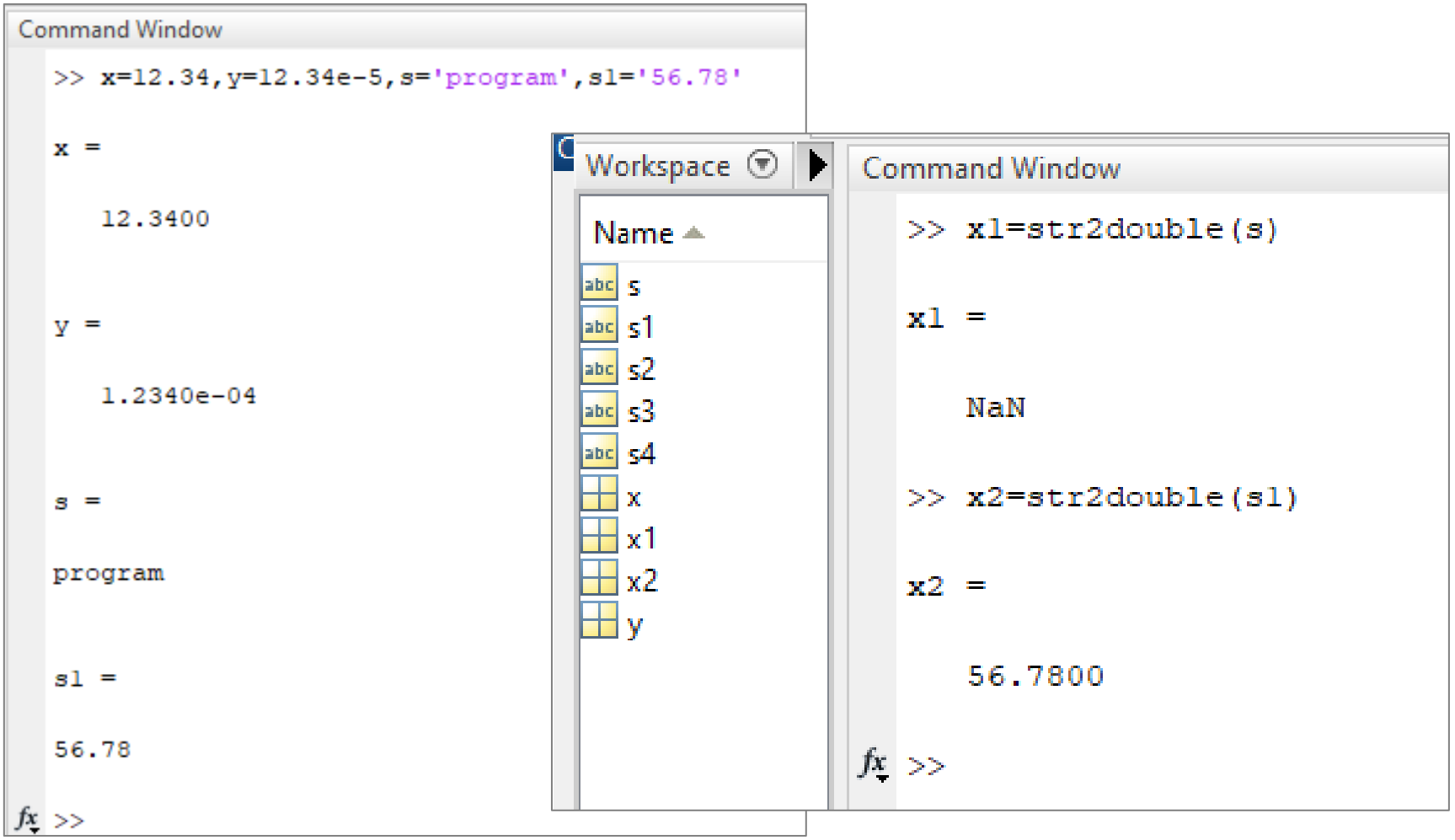
    %s – łańcuch znakowy

    %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



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

>>
```

Workspace

Name

- s
- s1
- s2
- s3
- s4
- x
- y

Command Window

```
>> s2=num2str(x)

s2 =

12.34

>> s3=int2str(x)

s3 =

12

>> s4=num2str(y)

s4 =

0.0001234

>>
```

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

>>
```

```
>> 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

- **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 wersalikami) – składnia: ***global ZMIENNE***
  - nazwa pliku z funkcją powinna być taka sama jak nazwa funkcji

Składnia funkcji

- Pierwsza linia m-pliku definiującego 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

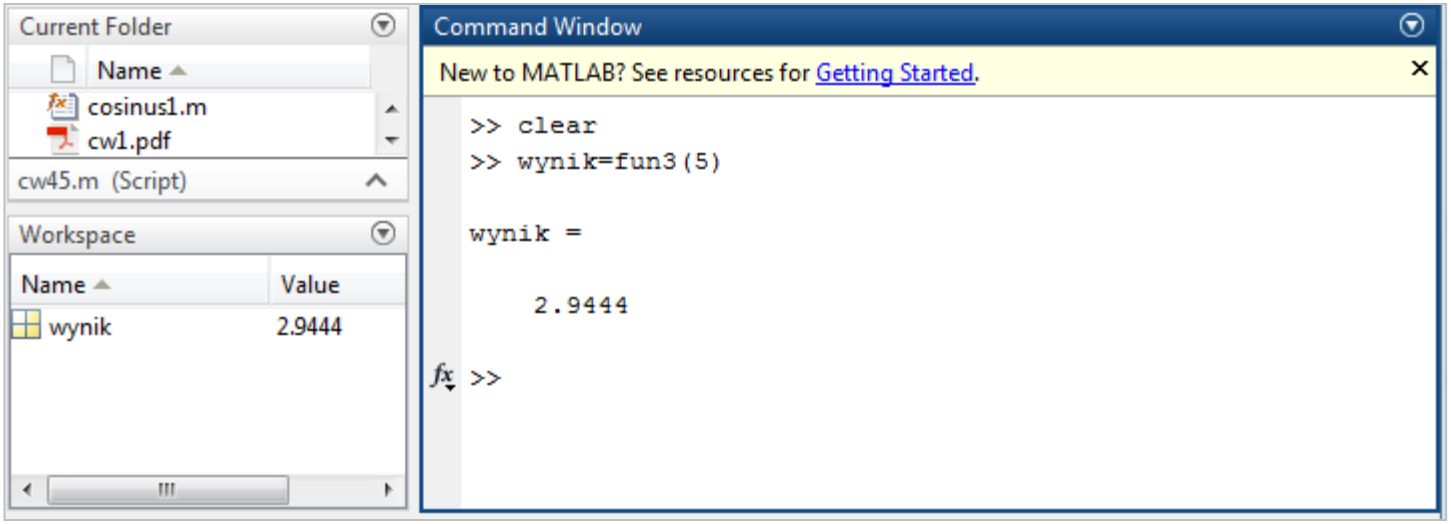
Editor - E:\zajęcia\Matlab\fun3.m

fun1.m | fun2.m | fun3.m

```
function [y]=fun3(x)
 n=length(x);
 y=ones(1,n);
 for i=1:n
    if x(i)>3
        y(i)=log(x(i)^2-6);
    elseif x<0
        y(i)=sqrt(x(i)^2/abs(x(i)-1));
    else
        y(i)=x(i);
    end
 end
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



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:

  function rad = radiany(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 wejść wyświetla komunikaty:

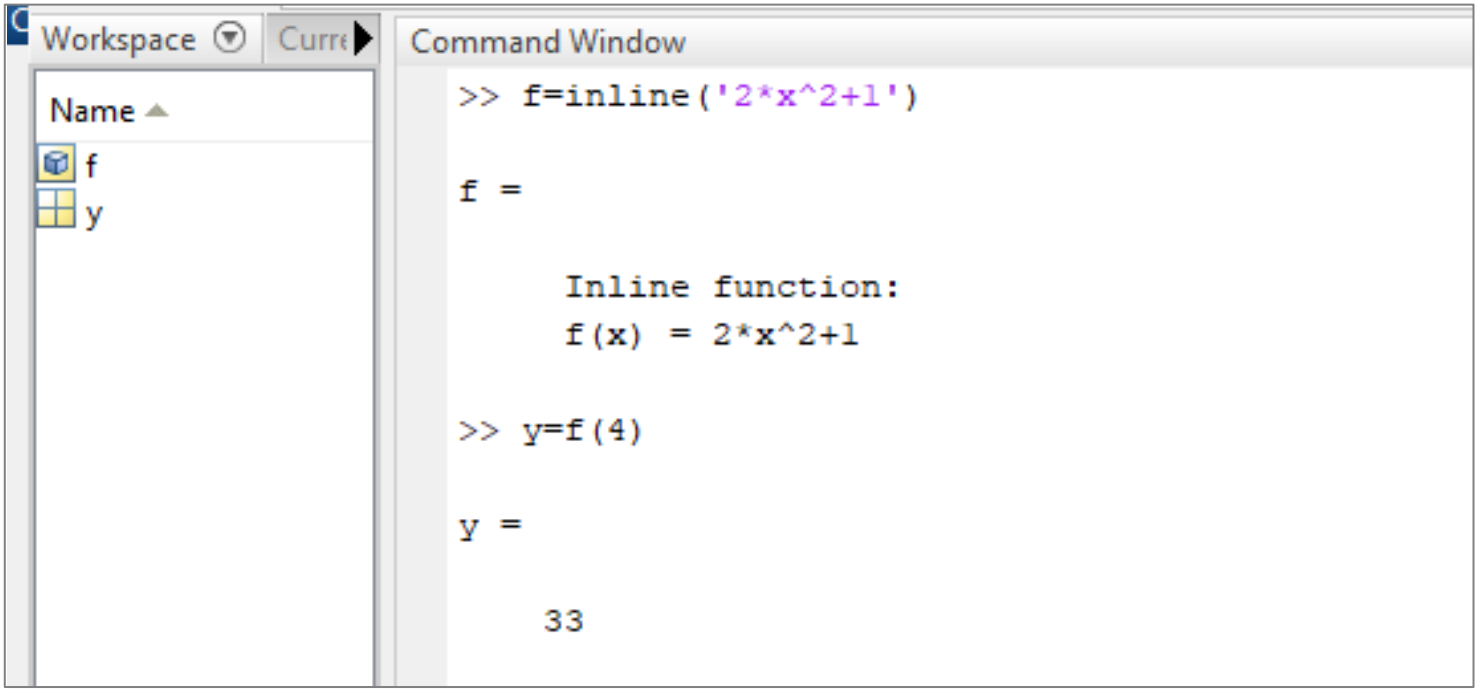
  'Not enough output arguments.'

  lub

  'Too many output arguments.'

Funkcja *inline*

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



Funkcja anonimowe

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