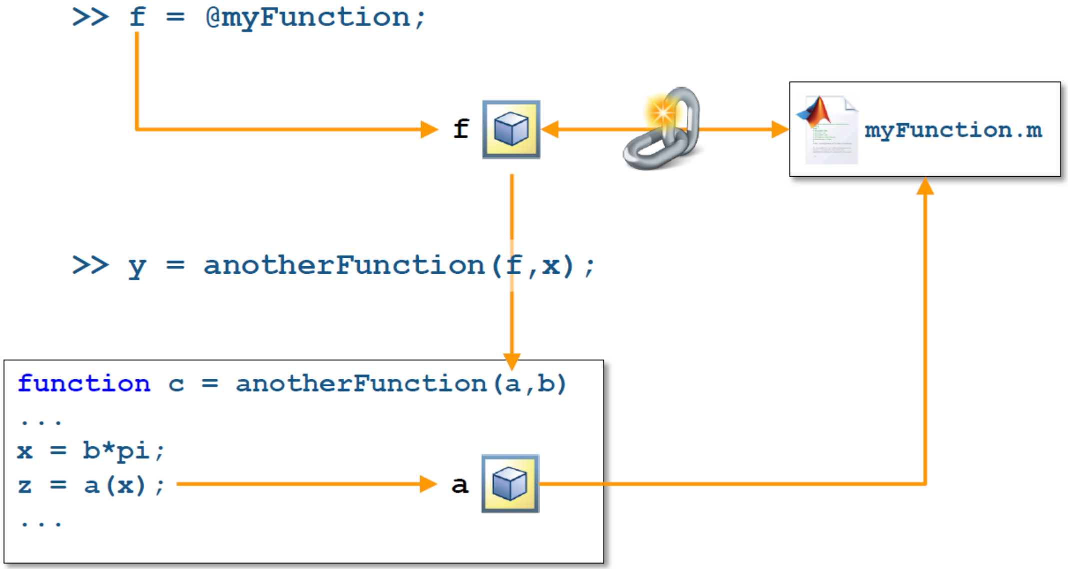
Function Handles

MATLAB® Programming Techniques

Duy NGUYEN
Engineering Development Group

What are function handles?



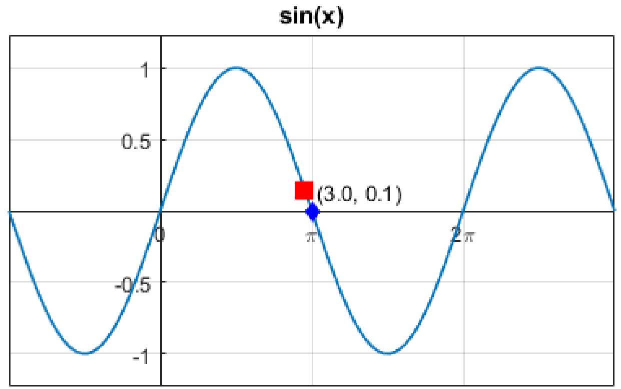
```
>> f = @myFunction;
```

```
>> y = anotherFunction(f,x);
```

```
function c = anotherFunction(a,b)
...
x = b*pi;
z = a(x);
...
```

What are function handles?

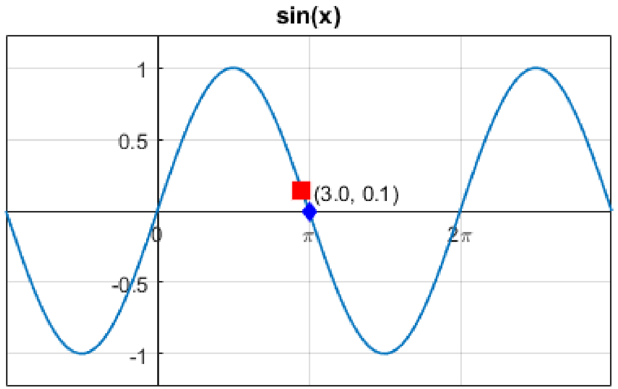
- Find a root of $\sin(x)$ near $x_0 = 3$:



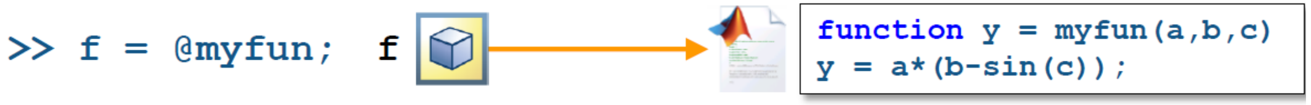
What are function handles?

- Find a root of $\sin(x)$ near $x_0 = 3$:

```
fun = @sin
x0 = 3
z = fzero(fun,x0)
```

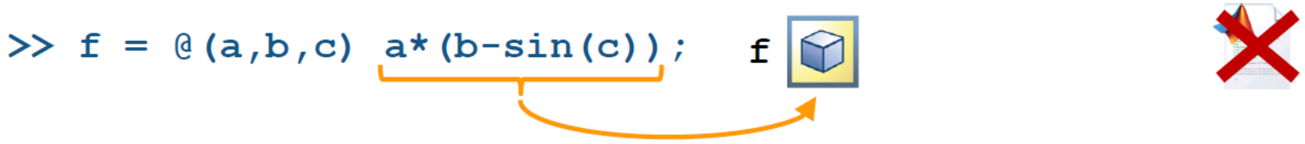


What are anonymous functions?



```
>> f = @myfun;
```

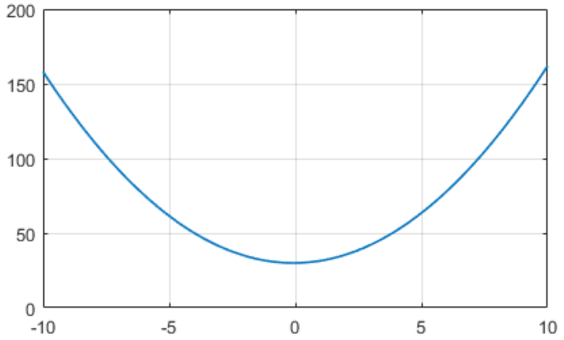
```
function y = myfun(a,b,c)
y = a*(b-sin(c));
```



```
>> f = @(a,b,c) a*(b-sin(c));
```

What are anonymous functions?

- Plot a parabola:

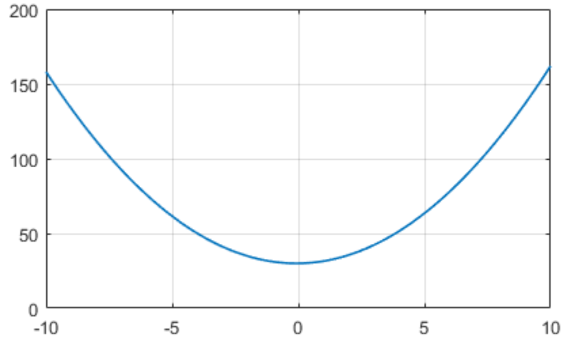


What are anonymous functions?

- Plot a parabola:

```
a = 1.3; b = 0.2; c = 30
myParabola = @(x) a*x.^2 + b*x + c;
x = linspace(-10, 10, 100)
plot(x, myParabola(x))
```

- Return information about a function handle:



What are anonymous functions?

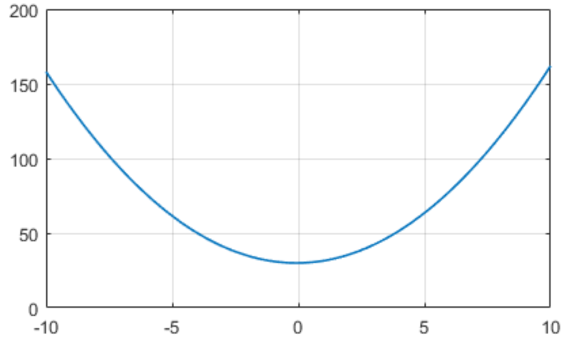
- Plot a parabola:

```
a = 1.3; b = 0.2; c = 30
myParabola = @(x) a*x.^2 + b*x + c;
x = linspace(-10, 10, 100)
plot(x, myParabola(x))
```

- Return information about a function handle:

```
info = functions(myParabola)
```

- Get the variables stored in the function:



What are anonymous functions?

- Plot a parabola:

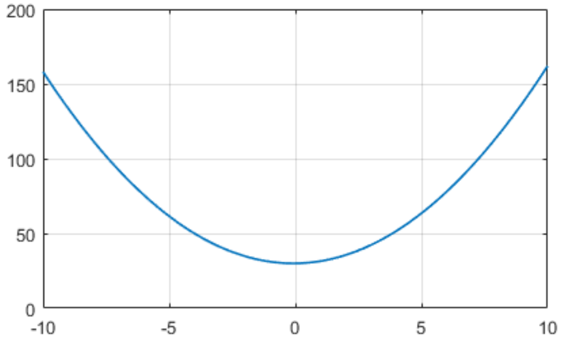
```
a = 1.3; b = 0.2; c = 30
myParabola = @(x) a*x.^2 + b*x + c;
x = linspace(-10, 10, 100)
plot(x, myParabola(x))
```

- Return information about a function handle:

```
info = functions(myParabola)
```

- Get the variables stored in the function:

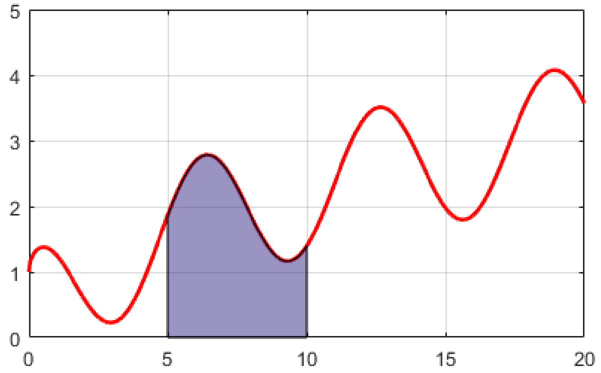
```
info.workspace{:}
```



What are anonymous functions?

- Compute the following definite integral

$$g(a, b) = \int_a^b \cos(x) + \sqrt{\frac{x}{2}}\,\mathrm{d}x$$

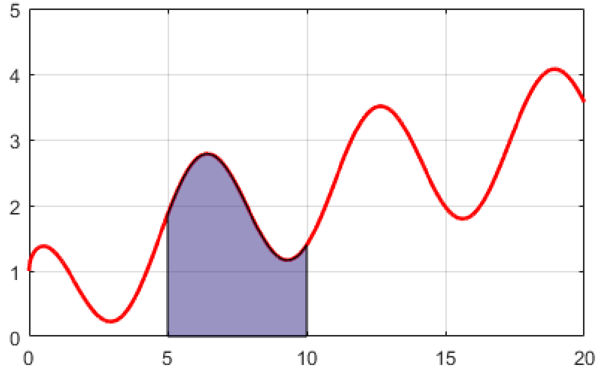


What are anonymous functions?

- Compute the following definite integral

$$g(a,b) = \int_a^b \cos(x) + \sqrt{\frac{x}{2}}\,\mathrm{d}x$$

```
fun = @(x) cos(x) + sqrt(x/2)
g = @(a, b) integral(fun, a, b)
g(5, 10)
```



Changing the interface with anonymous functions

```
function y = myfun(a,b,c)
y = a*(b-sin(c));
```

```
>> f = @myfun;
```

```
>> fzero(f,0.5)
```

Expects a function of
1 input argument

```
>> f = @wrapper;
```

```
function y = wrapper(b)
a = readparameters();
y = myfun(a,b,pi/4);
```

```
>> fzero(f,0.5)
```

```
f
```

1 input
argument

```
>> a = readparameters();
>> f = @(b) myfun(a,b,pi/4);
```

```
>> fzero(f,0.5)
```