Functions I

Functions I

A function is a piece of reusable code

Functions I

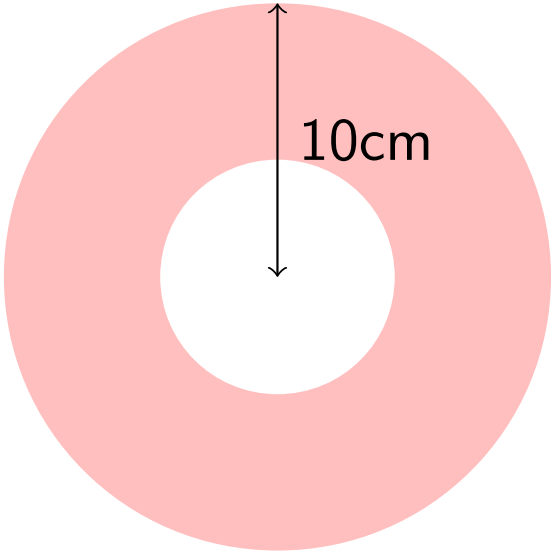
A function is a piece of reusable code
Recall the area of a doughnut...

Functions I

A function is a piece of reusable code
Recall the area of a doughnut...

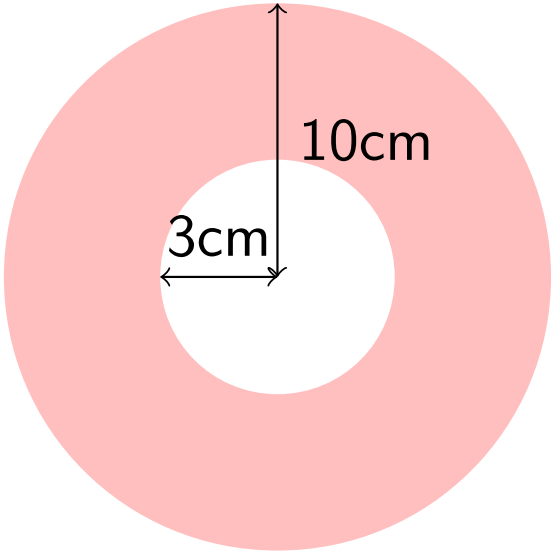
Functions I

A function is a piece of reusable code
Recall the area of a doughnut...



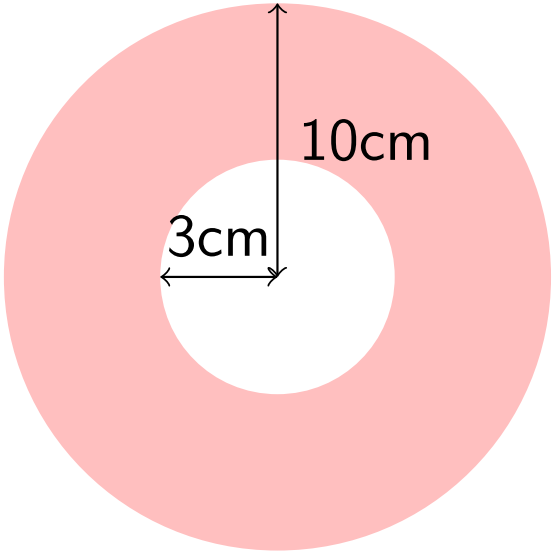
Functions I

A function is a piece of reusable code
Recall the area of a doughnut...



Functions I

A function is a piece of reusable code
Recall the area of a doughnut...

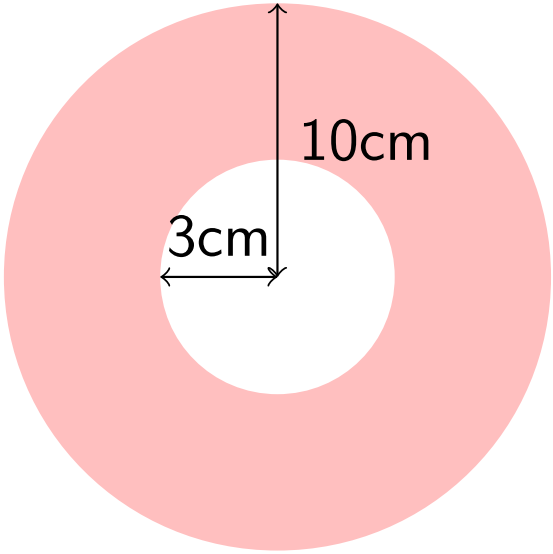


You could do this

pi∗10^2 − **pi**∗3^2

Functions I

A function is a piece of reusable code
Recall the area of a doughnut...



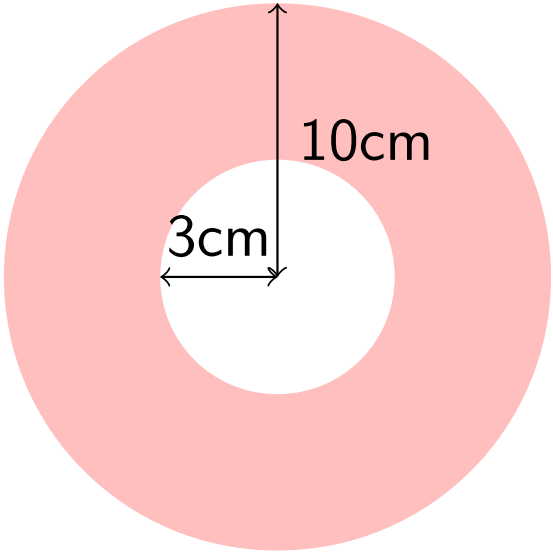
You could do this

pi∗10^2 − **pi**∗3^2

But this would be better

circle_area (10) − circle_area (3)

Functions I



A function is a piece of reusable code
Recall the area of a doughnut...

You could do this

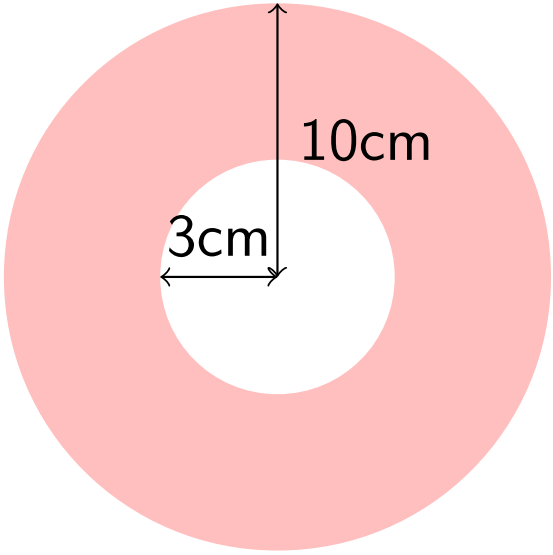
pi∗10^2 − **pi**∗3^2

But this would be better

circle_area (10) − circle_area (3)

Functions are used in place of their explicit *algorithms*

Functions I



A function is a piece of reusable code
Recall the area of a doughnut...

You could do this

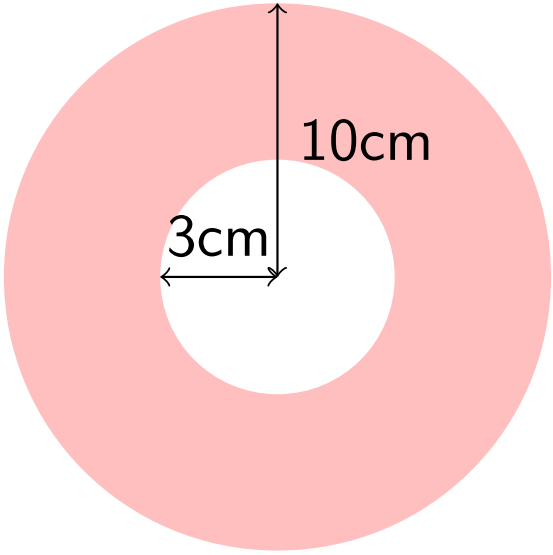
```
pi*10^2 - pi*3^2
```

But this would be better

```
circle_area (10) - circle_area (3)
```

Functions are used in place of their explicit *algorithms*
Some Terminology:

Functions I



A function is a piece of reusable code
Recall the area of a doughnut...

You could do this

pi∗10^2 − **pi**∗3^2

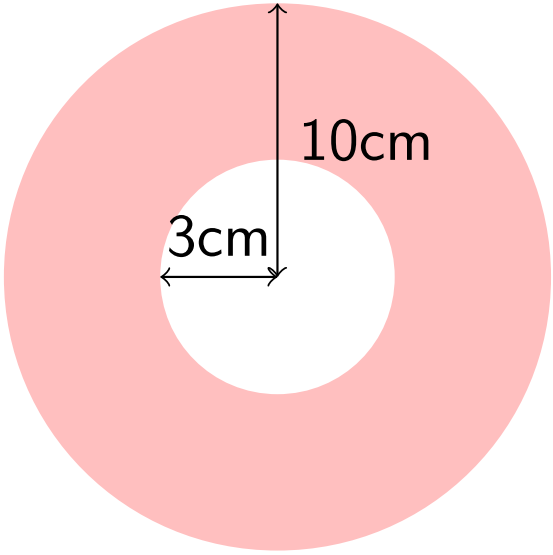
But this would be better

circle_area (10) − circle_area (3)

Functions are used in place of their explicit *algorithms*
Some Terminology:

- Functions are **call**ed "call circle_area"

Functions I



A function is a piece of reusable code
Recall the area of a doughnut...

You could do this

pi∗10^2 − **pi**∗3^2

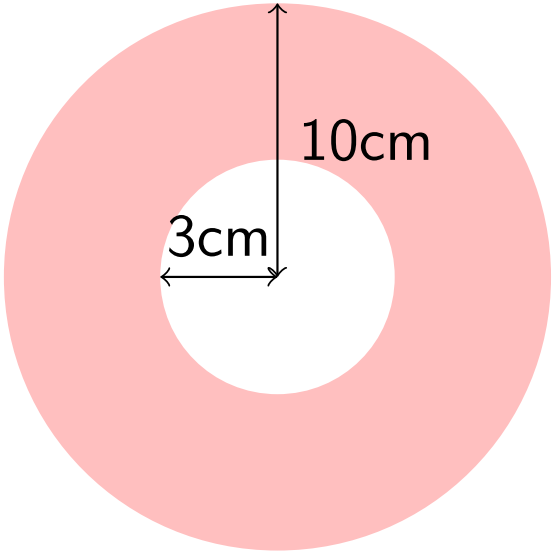
But this would be better

circle_area (10) − circle_area (3)

Functions are used in place of their explicit *algorithms*
Some Terminology:

- Functions are **call**ed — "call circle_area"
- Input parameters are **pass**ed — "pass 10"

Functions I



A function is a piece of reusable code
Recall the area of a doughnut...

You could do this

```
pi*10^2 - pi*3^2
```

But this would be better

```
circle_area (10) - circle_area (3)
```

Functions are used in place of their explicit *algorithms*
Some Terminology:

- Functions are **call**ed — "call circle_area"
- Input parameters are **pass**ed — "pass 10"
- Output values are **return**ed — "returns 314.159"

Builtin Functions

Builtin Functions

Most languages have already defined functions

Builtin Functions

Most languages have already defined functions
Matlab has a *lot* of builtin functions

Builtin Functions

Most languages have already defined functions
Matlab has a *lot* of builtin functions

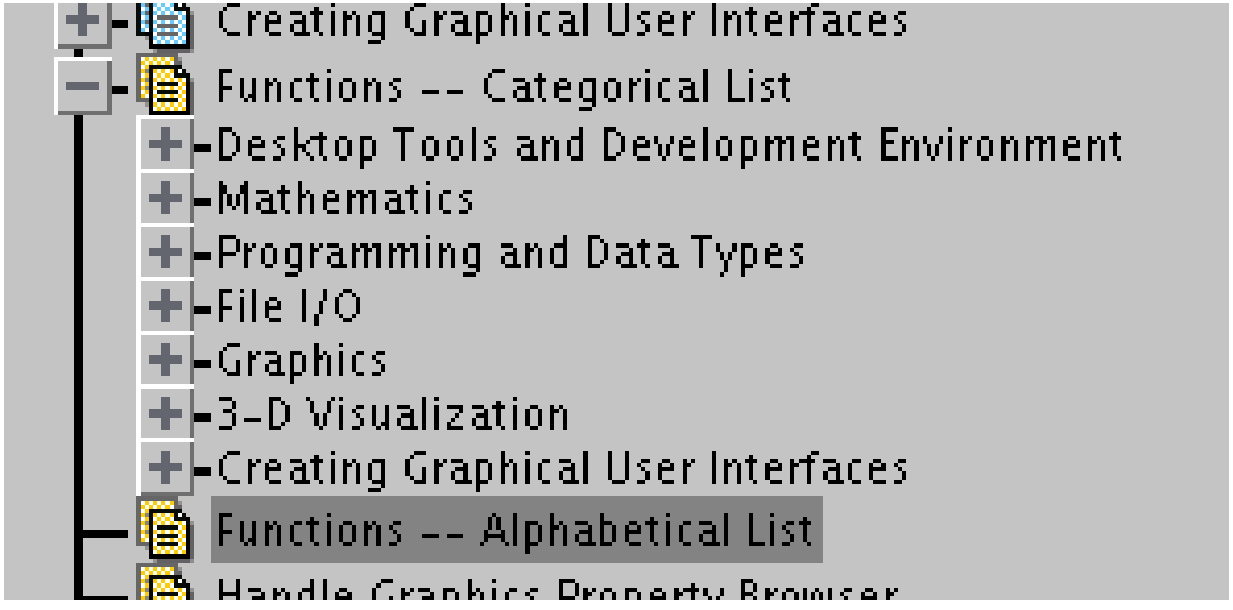
abs(), **max**(), **mean**(), **floor**()

Builtin Functions

Most languages have already defined functions
Matlab has a *lot* of builtin functions

abs(), **max**(), **mean**(), **floor**()

Your best reference is the Matlab Help system



Rolling your own

Rolling your own

- Most of the time, you want your own functions

Rolling your own

- Most of the time, you want your own functions
- Any serious language allows "user-defined functions"

Rolling your own

- Most of the time, you want your own functions
- Any serious language allows "user-defined functions"
- Matlab uses "m-files" to contain user-defined functions

Rolling your own

- Most of the time, you want your own functions
- Any serious language allows "user-defined functions"
- Matlab uses "m-files" to contain user-defined functions

Possible contents of `circle_area.m`

```
function area = circle_area( radius )
% function area = circle_area( radius )
%
% Computes the area of a circle

area = pi * radius^2
```

Rolling your own

- Most of the time, you want your own functions
- Any serious language allows "user-defined functions"
- Matlab uses "m-files" to contain user-defined functions

Possible contents of `circle_area.m`

```
function area = circle_area( radius )
% function area = circle_area( radius )
%
% Computes the area of a circle

area = pi * radius^2
```

Let's unpack it...

Anatomy of a Function

```
function area = circle_area( radius )
% function area = circle_area( radius )
%
% Computes the area of a circle

area = pi * radius^2
```

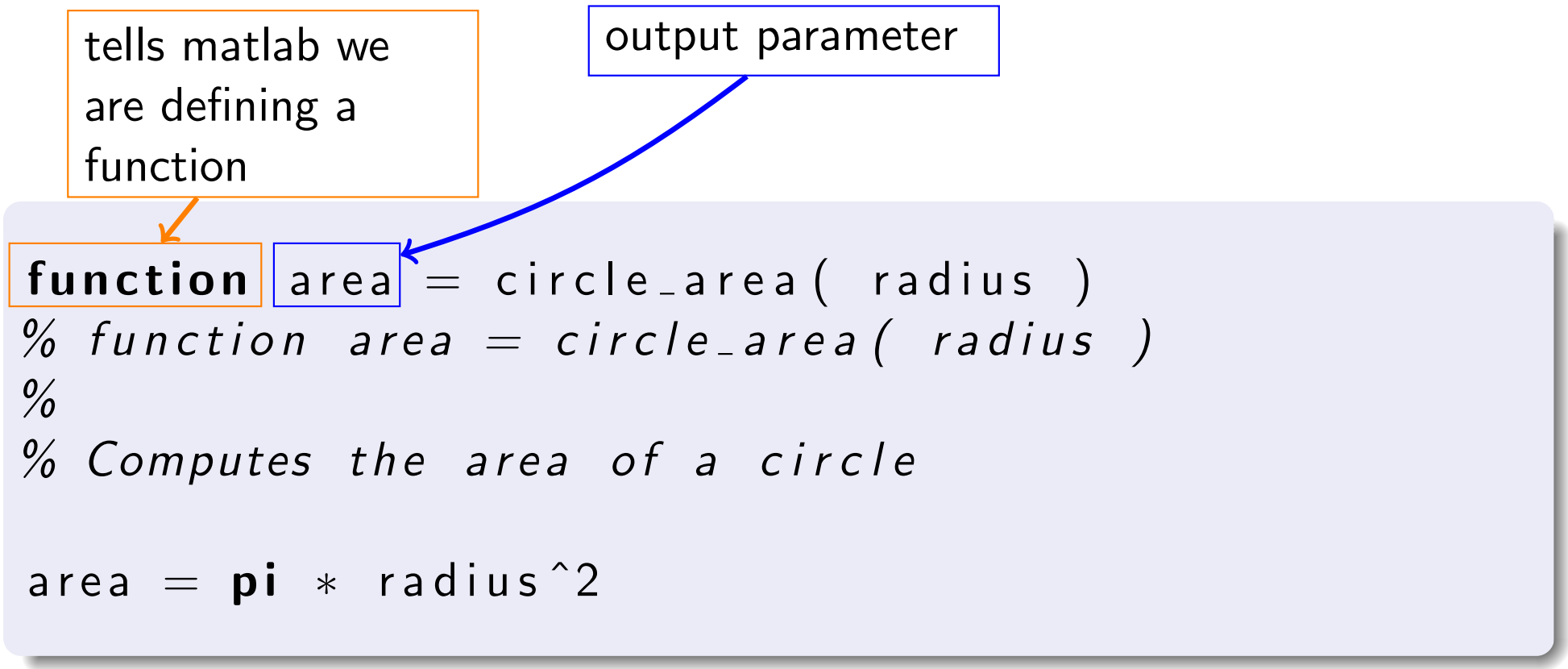
Anatomy of a Function

tells matlab we are defining a function

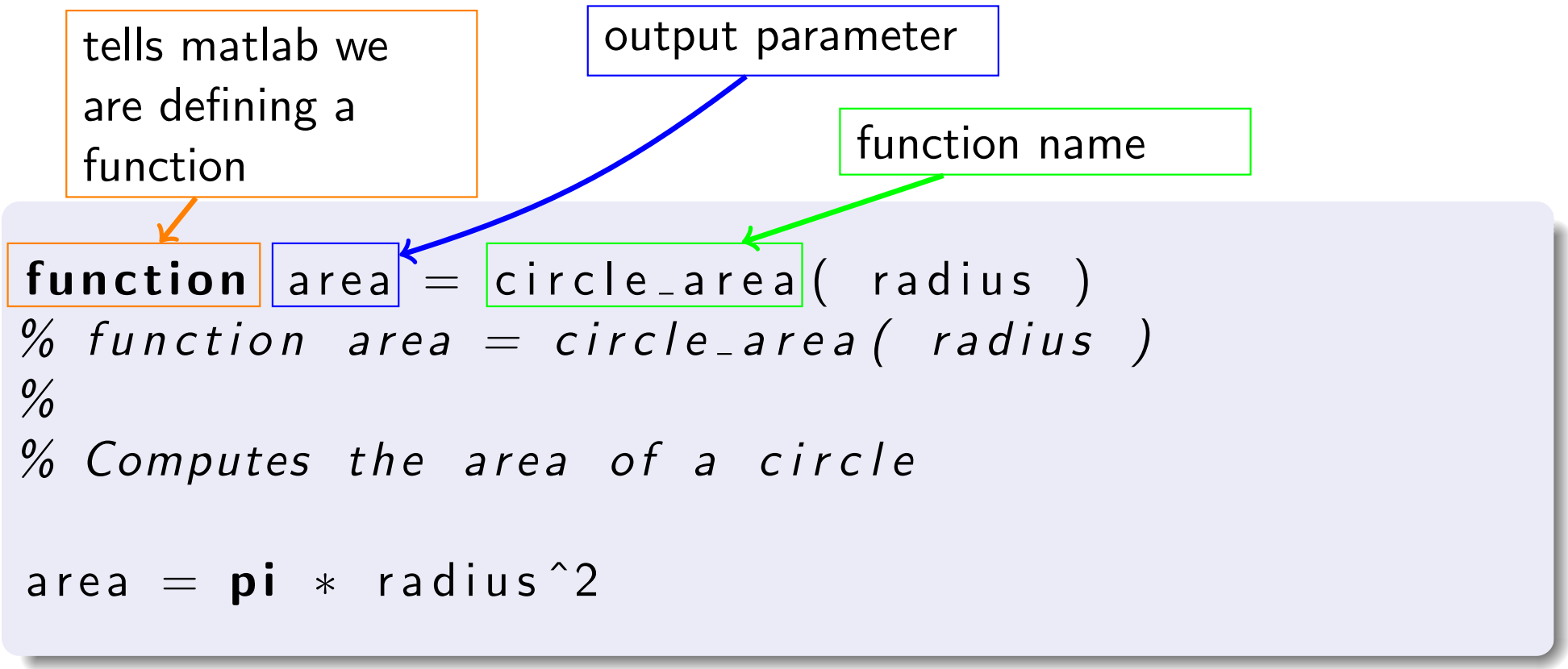
```
function area = circle_area( radius )
% function area = circle_area( radius )
%
% Computes the area of a circle

area = pi * radius^2
```

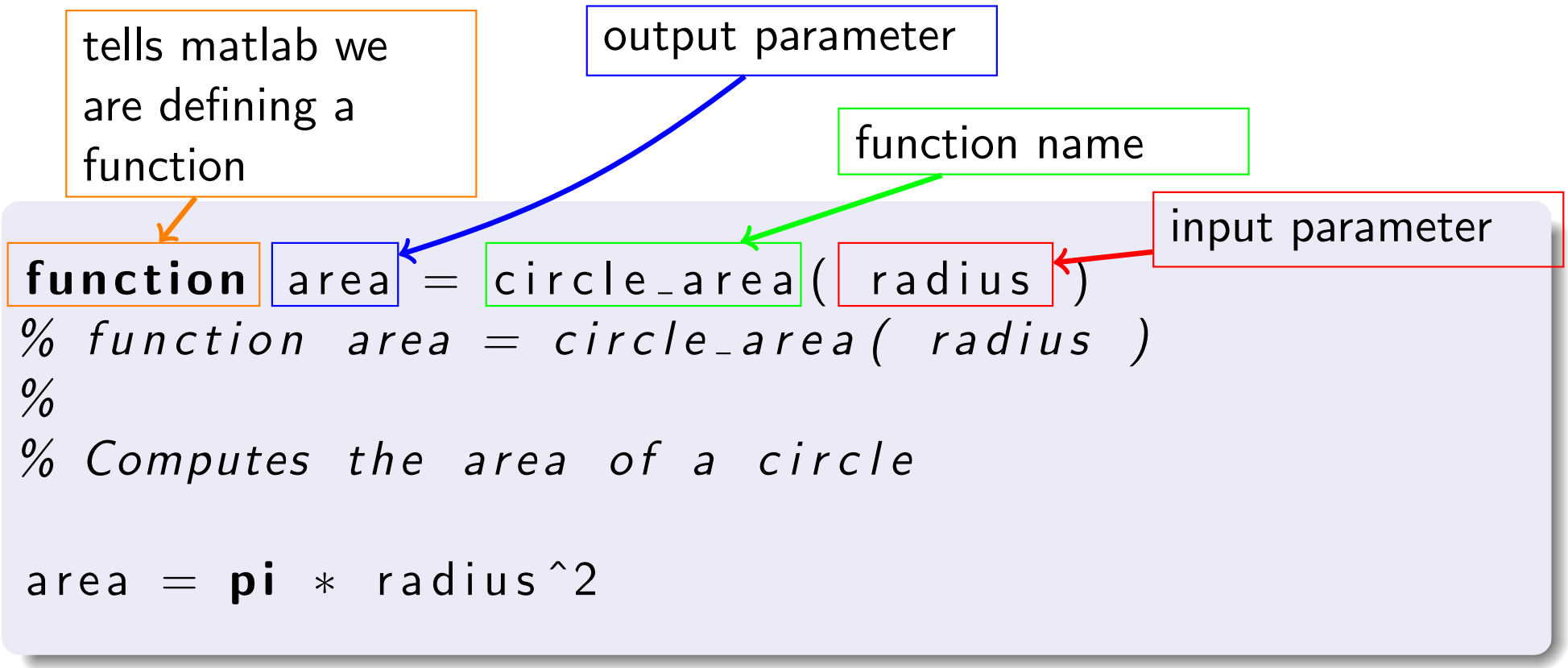
Anatomy of a Function



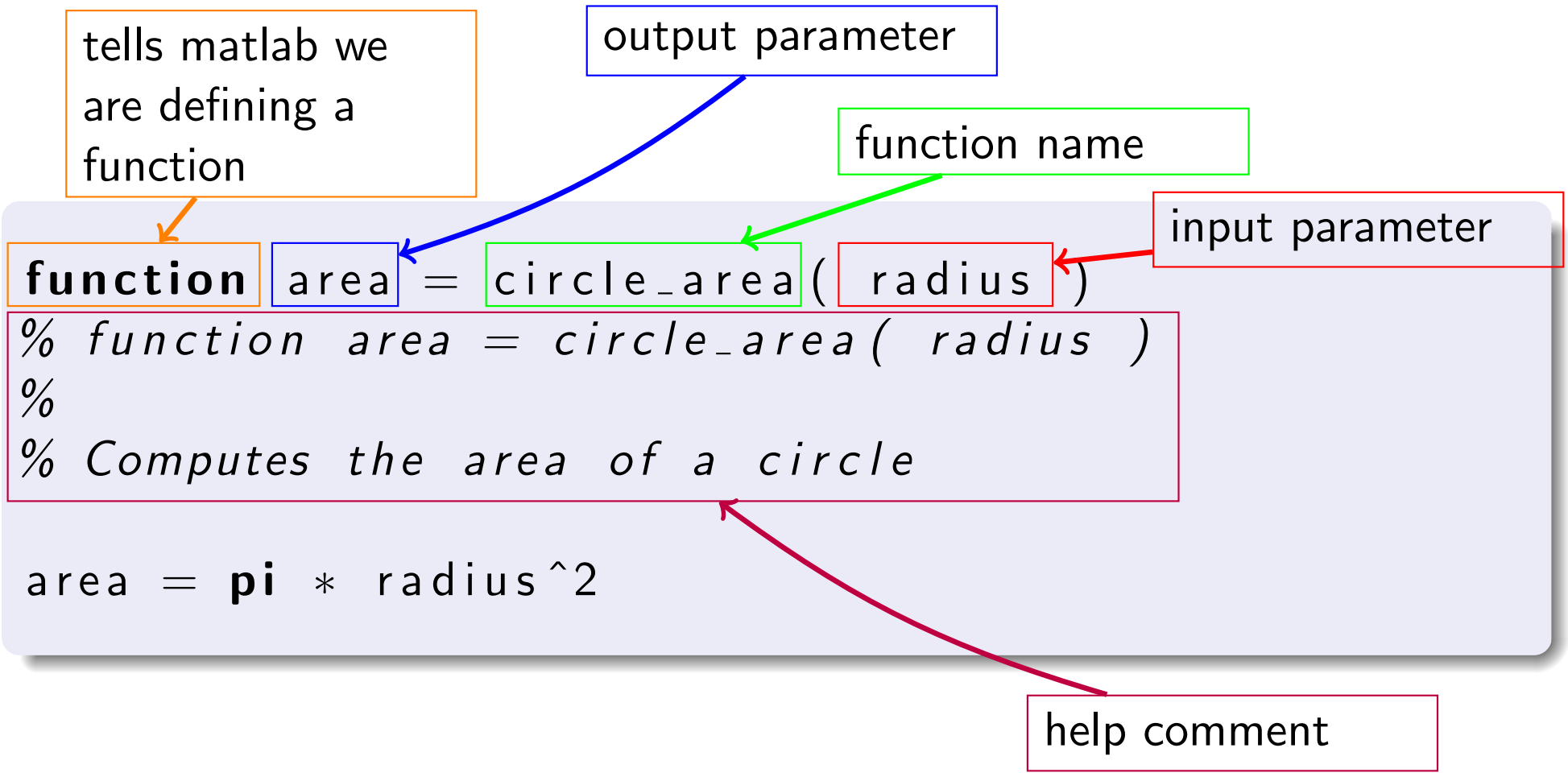
Anatomy of a Function



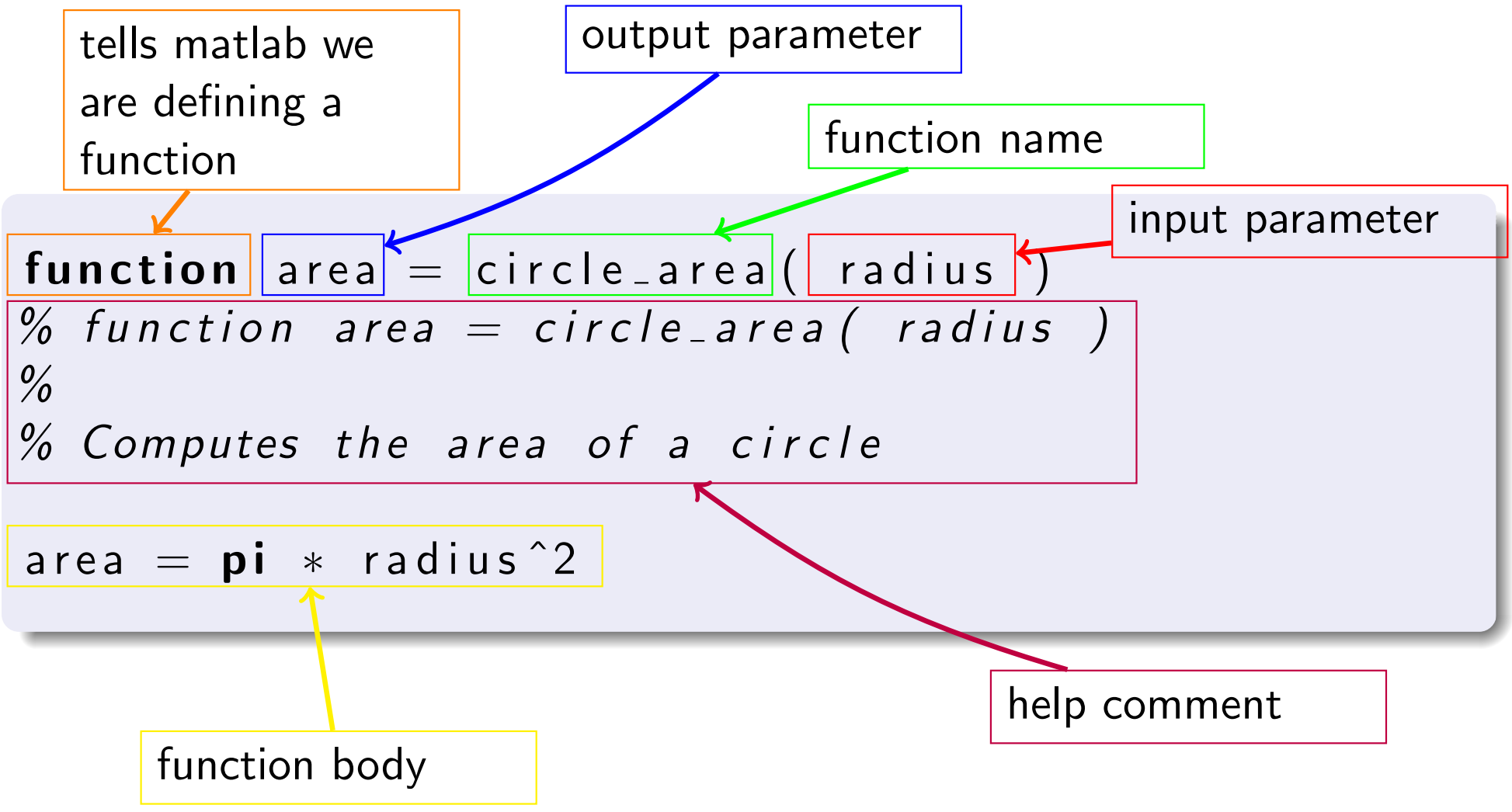
Anatomy of a Function



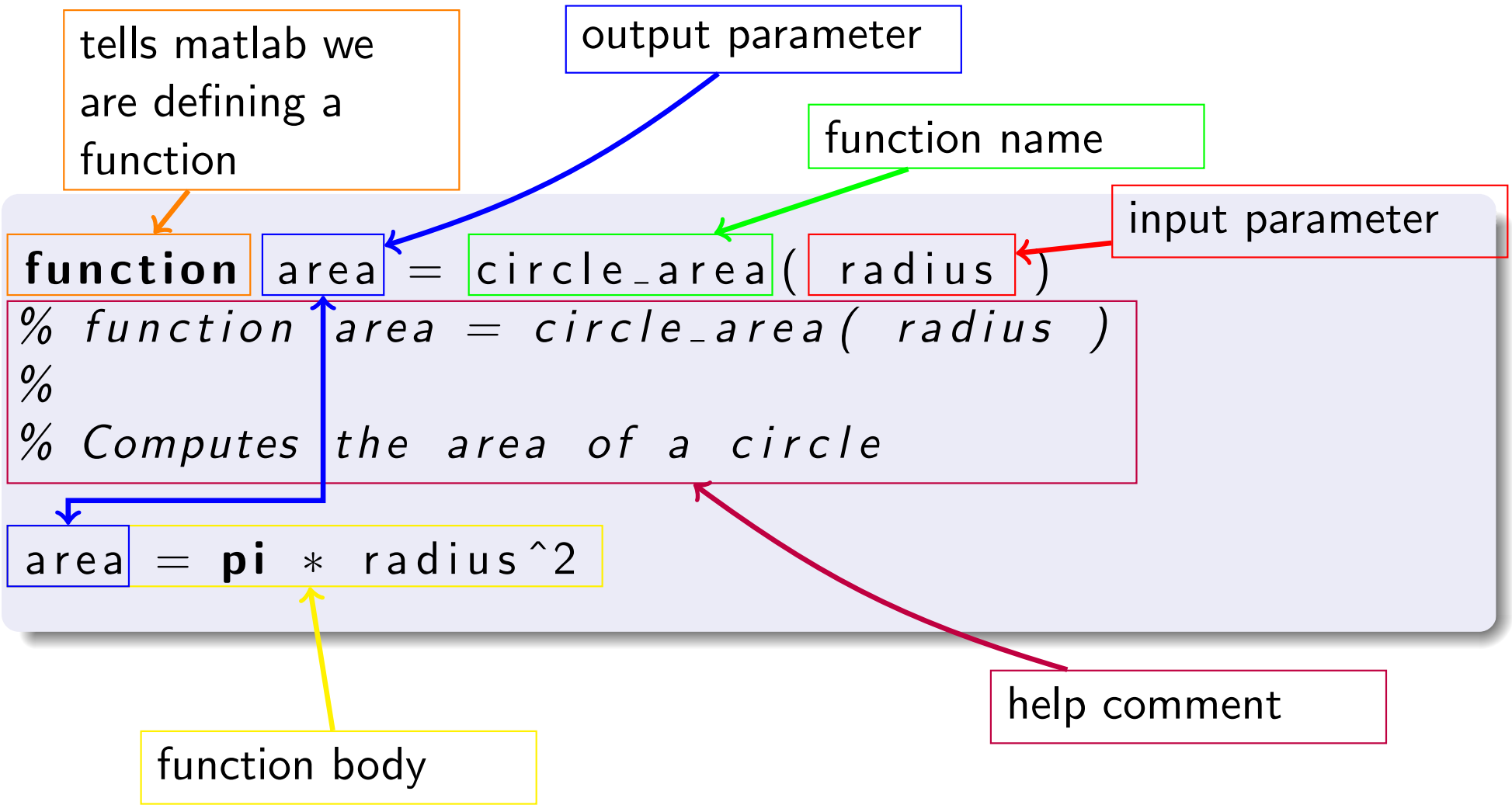
Anatomy of a Function



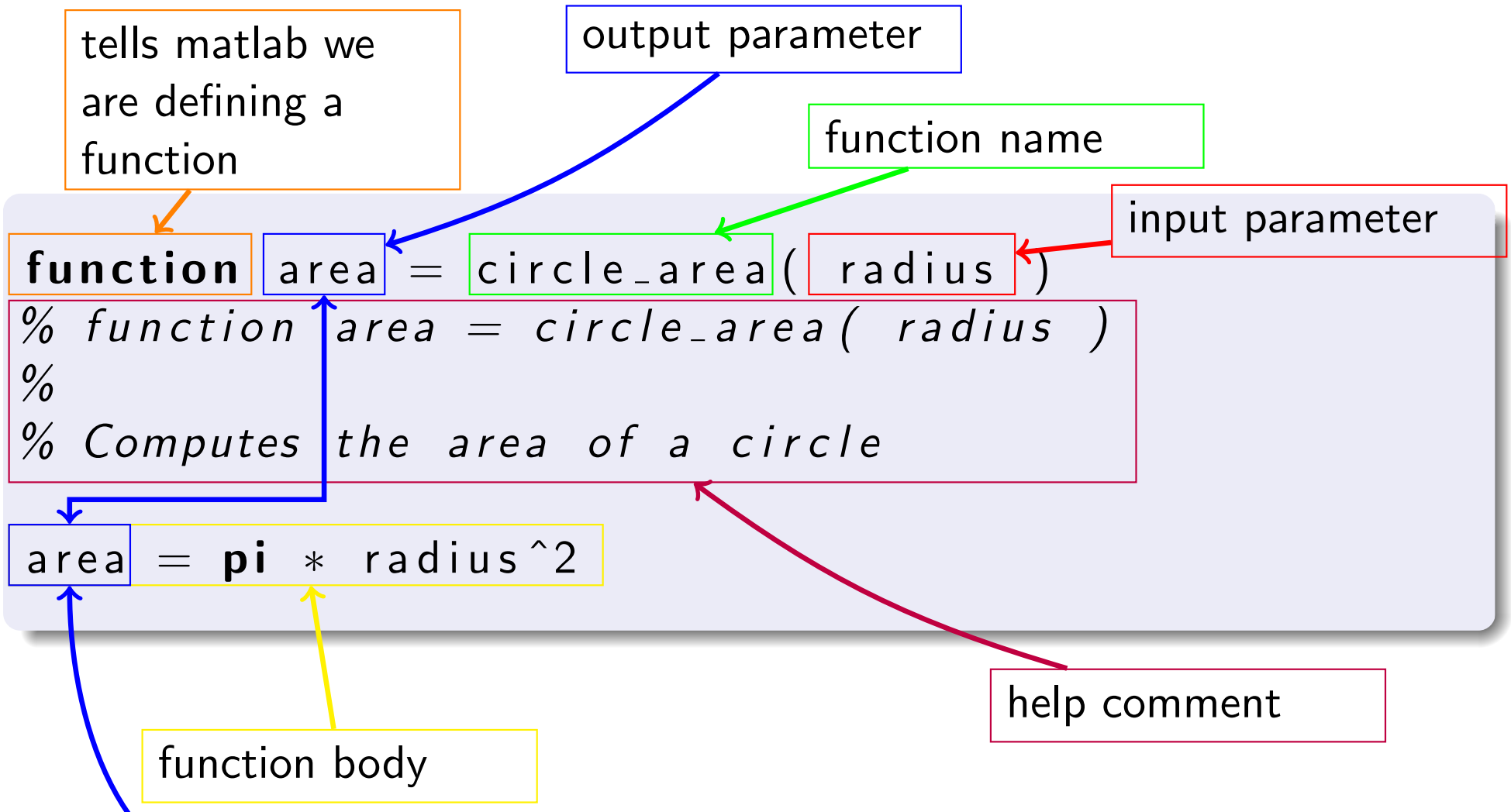
Anatomy of a Function



Anatomy of a Function



Anatomy of a Function



Other Languages
Common

C

```c
float circle_area(float radius)
{
    float area;

    area = MATH_PI * radius * radius;

    return area;
}
```

Python

```python
def circle_area(radius):
    area = math.pi * radius^2

    return area
```

Other Languages
Less Common

FORTRAN

```
      real function circle_area(radius)
c Compute the area of a circle
      real radius
      parameter(pi = 3.14159)
      circle_area = pi * radius**2
      return
```

Haskell

```
circle_area :: Num a => a -> a
circle_area radius = pi * radius^2
```