

CS100: CPADS

Functions & Function Calls

David Babcock / Don Hake
Department of Physical Sciences
York College of Pennsylvania



Function Concepts

- **Encapsulation**

- Modularize commonly used code that performs a specific task

- **Generalization**

- Perform a similar operation on different values that are provided via **parameters**

- **Functions can be executed via a function call**

- The function call provides **arguments** to the function
- The **arguments** correspond to **parameters** that are declared in the **function definition**

Python's Function Syntax

- **All function definitions in Python must:**

- Start with the `def` keyword
 - This is the start of the function declaration
- Have a function name
- Have a parameter list (even if it's empty)
- Include one or more INDENTED statements as the function body


```
def function_name(parameter_list, ... , ... ):
    statement1
    statement2
    etc.
```

Functions & Function Calls: Example

```
1 # Load TurtleWorld functions
2 from swampy.TurtleWorld import *
3
4 def main():
5     world = TurtleWorld()      # Create TurtleWorld object
6     turtle = Turtle()          # Create Turtle object
7
8     # Draw graphics
9     fd(turtle, 100)
10    rt(turtle, 90)
11    fd(turtle, 100)
12    rt(turtle, 90)
13
14    # Press enter to exit
15    key = raw_input('Press enter to exit')
16    world.destroy()
17
18 main()
```

NOTE: A program's flow is not necessarily linear

Functions & Function Calls: Example



```
1 # Load TurtleWorld functions
2 from swampy.TurtleWorld import *
3
4 def main():
5     world = TurtleWorld()           # Create TurtleWorld object
6     turtle = Turtle()               # Create Turtle object
7
8     # Draw graphics
9     fd(turtle,100)
10    rt(turtle,90)
11    fd(turtle,100)
12    rt(turtle,90)
13
14    # Press enter to exit
15    key = raw_input('Press enter to exit')
16    world.destroy()
17
18 main()
```

1

Run line 2, it is not part of a function

Functions & Function Calls: Example

```
1 # Load TurtleWorld functions
2 from swampy.TurtleWorld import *
```

1

2

Run line 4, find the function declaration for main().
DO NOT EXECUTE THE CODE INSIDE the main function yet.

```
3
4 def main():
5     world = TurtleWorld()      # Create TurtleWorld object
6     turtle = Turtle()          # Create Turtle object
7
8     # Draw graphics
9     fd(turtle,100)
10    rt(turtle,90)
11    fd(turtle,100)
12    rt(turtle,90)
13
14    # Press enter to exit
15    key = raw_input('Press enter to exit')
16    world.destroy()
17
18 main()
```

Functions & Function Calls: Example

```
1 # Load TurtleWorld functions
2 from swampy.TurtleWorld import *
3
4 def main():
5     world = TurtleWorld() # Create TurtleWorld object
6     turtle = Turtle()     # Create Turtle object
7
8     # Draw graphics
9     fd(turtle,100)
10    rt(turtle,90)
11    fd(turtle,100)
12    rt(turtle,90)
13
14    # Press enter to exit
15    key = raw_input('Press enter to exit')
16    world.destroy()
17
18 main()
```

Note that the `main()` function is indented. The end of the **function definition** is where the indentation stops.

Functions & Function Calls: Example

```
1 # Load TurtleWorld functions
2 from swampy.TurtleWorld import *
3
4 def main():
5     world = TurtleWorld()           # Create TurtleWorld object
6     turtle = Turtle()               # Create Turtle object
7
8     # Draw graphics
9     fd(turtle, 100)
10    rt(turtle, 90)
11    fd(turtle, 100)
12    rt(turtle, 90)
13
14    # Press enter to exit
15    key = raw_input('Press enter to exit')
16    world.destroy()
```

➔ 18 `main()` **3** Run line 18. Find the **function call** to the previously defined `main()` function.

I remember seeing a function named `main` on line 4



Functions & Function Calls: Example

```
1 # Load TurtleWorld functions
2 from swampy.TurtleWorld import *
3
4 def main():
5     world = TurtleWorld()
6     turtle = Turtle()
7
8     # Draw graphics
9     fd(turtle,100)
10    rt(turtle,90)
11    fd(turtle,100)
12    rt(turtle,90)
13
14    # Press enter to exit
15    key = raw_input('Press enter to exit')
16    world.destroy()
17
18 main()
```

1

2

4

The program **calls** the `main()` function on line 4 and continues execution on line 5

3

Functions & Function Calls: Example

```
1 # Load TurtleWorld functions
2 from swampy.TurtleWorld import *
3
4 def main():
5     world = TurtleWorld()
6     turtle = Turtle()
7
8     # Draw graphics
9     fd(turtle, 100)
10    rt(turtle, 90)
11    fd(turtle, 100)
12    rt(turtle, 90)
13
14    # Press enter to exit
15    key = raw_input('Press enter to exit')
16    world.destroy()
17
18 main()
```

1

2

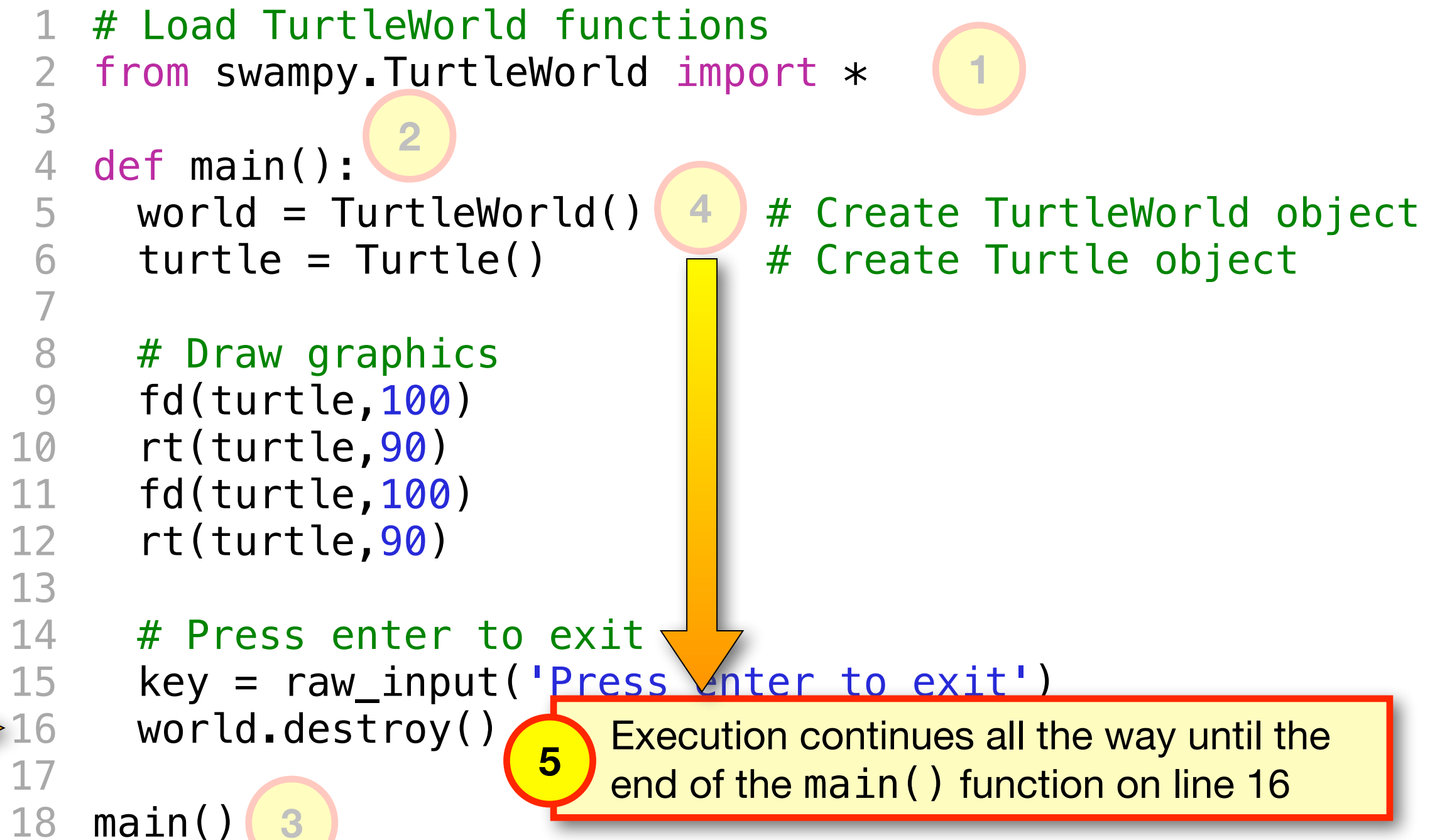
4

Create TurtleWorld object
Create Turtle object

5

Execution continues all the way until the end of the main() function on line 16

3



Functions & Function Calls: Example

```
1 # Load TurtleWorld functions
2 from swampy.TurtleWorld import *
3
4 def main():
5     world = TurtleWorld()
6     turtle = Turtle()
7
8     # Draw graphics
9     fd(turtle, 100)
10    rt(turtle, 90)
11    fd(turtle, 100)
12    rt(turtle, 90)
13
14    # Press enter to exit
15    key = raw_input('Press enter to exit')
16    world.destroy()
17
18 main()
```

6

After executing the body of the main function,
execution resumes after the point of call (line 19)

Function Calls

```
1 # Load TurtleWorld functions
2 from swampy.TurtleWorld import *
3
4 def main():
5     world = TurtleWorld()           # Create TurtleWorld object
6     turtle = Turtle()               # Create Turtle object
7
8     # Draw graphics
9     fd(turtle,100)
10    rt(turtle,90)
11    fd(turtle,100)
12    rt(turtle,90)
13
14    # Press enter to exit
15    key = raw_input('Press enter to exit')
16    world.destroy()
17
18    main()
```

Guess what? You've been
using LOTS of function calls
already!



Function Calls

```
1 # Load TurtleWorld functions
2 from swampy.TurtleWorld import *
3
4 def main():
5     world = TurtleWorld()           # Create TurtleWorld object
6     turtle = Turtle()               # Create Turtle object
7
8     # Draw graphics
9     fd(turtle, 100)
10    rt(turtle, 90)
11    fd(turtle, 100)
12    rt(turtle, 90)
13
14    # Press enter to exit
15    key = raw_input('Press enter to exit')
16    world.destroy()
17
18 main()
```

Some function calls don't require any arguments



Function Calls

```
1 # Load TurtleWorld functions
2 from swampy.TurtleWorld import *
3
4 def main():
5     world = TurtleWorld()           # Create TurtleWorld object
6     turtle = Turtle()               # Create Turtle object
7
8     # Draw graphics
9     fd(turtle,100)
10    rt(turtle,90)
11    fd(turtle,100)
12    rt(turtle,90)
13
14    # Press enter to exit
15    key = raw_input('Press enter to exit')
16    world.destroy()
17
18 main()
```

Other **function calls** may require
one or more **arguments**



Function Calls with Arguments

```
1 from swampy.TurtleWorld import *
2
3 def right_ang(t, size):
4     fd(t,size)
5     rt(t,90)
6     fd(t,size)
7     rt(t,90)
8
9 def main():
10     world = TurtleWorld()      # Create TurtleWorld object
11     turtle = Turtle()          # Create Turtle object
12     length = 100               # Define a length variable
13
14     right_ang(turtle, length)  # Call the right_ang function
15
16     # Press enter to exit
17     key = raw_input('Press enter to exit')
18     world.destroy()
19
20 main()
```

Here's another example that shows a function that requires two arguments

Function Calls with Arguments

```
1 from swampy.TurtleWorld import *
2
3 def right_ang(t, size):
4     fd(t,size)
5     rt(t,90)
6     fd(t,size)
7     rt(t,90)
8
9 def main():
10     world = TurtleWorld()           # Create TurtleWorld object
11     turtle = Turtle()               # Create Turtle object
12     length = 100                    # Define a length variable
13
14     right_ang(turtle, length) # Call the right_ang function
15
16     # Press enter to exit
17     key = raw_input('Press enter to exit')
18     world.destroy()
19
20 main()
```

Function declaration for
right_ang() function

Function Calls with Arguments

```
1 from swampy.TurtleWorld import *
2
3 def right_ang(t, size):
4     fd(t,size)
5     rt(t,90)
6     fd(t,size)
7     rt(t,90)
8
9 def main():
10     world = TurtleWorld()           # Create TurtleWorld object
11     turtle = Turtle()                # Create Turtle object
12     length = 100                     # Define a length variable
13
14     right_ang(turtle, length)        # Call the right_ang function
15
16     # Press enter to exit
17     key = raw_input('Press enter to exit')
18     world.destroy()
19
20 main()
```

Includes two **parameters**,
t and size that are only available
inside the right_ang() function

Function Calls with Arguments

```
1 from swampy.TurtleWorld import *
2
3 def right_ang(t, size):
4     fd(t,size)
5     rt(t,90)
6     fd(t,size)
7     rt(t,90)
8
9 def main():
10     world = TurtleWorld()           # Create TurtleWorld object
11     turtle = Turtle(world)          # Create Turtle object
12     length = 100                     # length variable
13
14     right_ang(turtle, length)        # Call the right_ang function
15
16     # Press enter to exit
17     key = raw_input('Press enter to exit')
18     world.destroy()
19
20 main()
```

The length variable is
assigned a value of 100

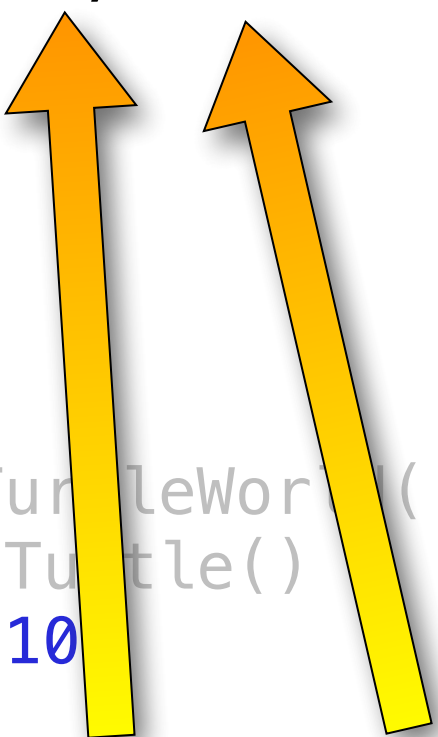
Function Calls with Arguments

```
1 from swampy.TurtleWorld import *
2
3 def right_ang(t, size):
4     fd(t,size)
5     rt(t,90)
6     fd(t,size)
7     rt(t,90)
8
9 def main():
10     world = TurtleWorld()           # Create TurtleWorld object
11     turtle = Turtle()               # Create Turtle object
12     length = 100                    # Define a length variable
13
14     right_ang(turtle, length)
15
16     # Press enter to exit
17     key = raw_input('Press enter to exit')
18     world.destroy()
19
20 main()
```

The length variable is passed as an argument to the right_ang() function

Function Calls with Arguments

```
1 from swampy.TurtleWorld import *
2
3 def right_ang(t, size):
4     fd(t,size)
5     rt(t,90)
6     fd(t,size)
7     rt(t,90)
8
9 def main():
10     world = TurtleWorld()
11     turtle = Turtle()
12     length = 10
13
14     right_ang(turtle, length) # Call the right_ang function
15
16     # Press enter to exit
17     key = raw_input('Press enter to exit')
18     world.destroy()
19
20 main()
```

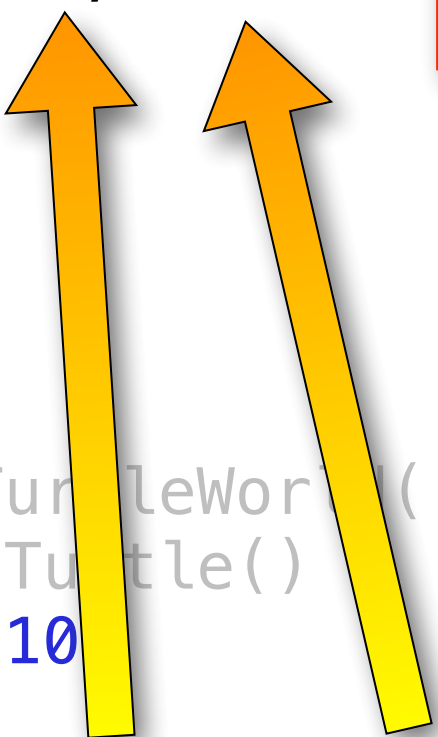


The values stored in the turtle variable and the length variable are given to the right_ang() function.
NOTE: The order matters.

Function Calls with Arguments

```
1 from swampy.TurtleWorld import *
2
3 def right_ang(t, size):
4     fd(t,size)
5     rt(t,90)
6     fd(t,size)
7     rt(t,90)
8
9 def main():
10     world = TurtleWorld()           # Create TurtleWorld object
11     turtle = Turtle()               # Create Turtle object
12     length = 100                    # Define a length variable
13
14     right_ang(turtle, length)        # Call the right_ang function
15
16     # Press enter to exit
17     key = raw_input('Press enter to exit')
18     world.destroy()
19
20 main()
```

The value of size becomes 100 since that was the value of length on line 14.



Function Calls with Arguments

```
1 from swampy.TurtleWorld import *
2
3 def right_ang(t, size):
4     fd(t,size)
5     rt(t,90)
6     fd(t,size)
7     rt(t,90)
8
9 def main():
10     world = TurtleWorld()           # Create TurtleWorld object
11     turtle = Turtle()                # Create Turtle object
12     length = 100                     # Define a length variable
13
14     right_ang(turtle, length) # Call the right_ang function
15
16     # Press enter to exit
17     key = raw_input('Press enter to exit')
18     world.destroy()
19
20 main()
```

When the right_ang() function is finished, the variables t and size disappear and **NO LONGER EXIST**

Terms to Remember

- **Function Declaration** - the first line of a function definition that includes the function name, and the list of parameters
- **Parameters** - the list of variables that a function expects to be provided when that function is called. These are included as part of the **function declaration**.
- **Function Body** - the body of a function is where all the work is done inside that function
- **Function Call** - a piece of code that redirects a program to execute code in a previously defined function. A function call **MUST** include a list of **arguments** that are passed into the **parameters** of the function.
- **Arguments** - literals, variables, or expressions that are included in a function call to provide information to a function