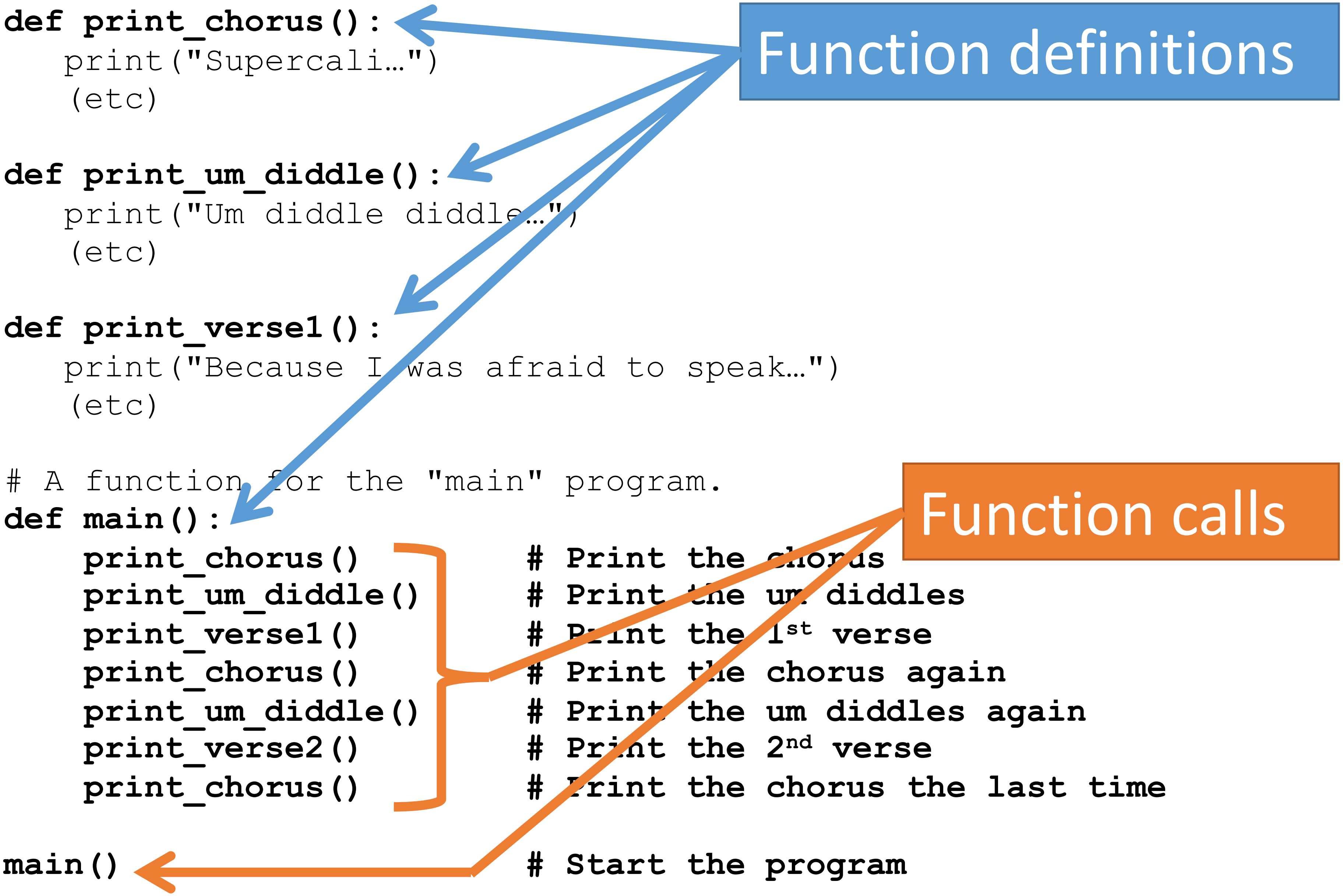
Functions with Parameters and Arguments

- We are using a lot of new files today, and we'll make a central folder to contain all of them.
- On the Windows desktops, double click on your student folder (there should be a link on the desktop to a folder than is named your name).
  - If you're using a personal computer, go to wherever you save your schoolwork.
- Make a new folder inside your student folder called **cs141**.
- In a web browser, go to cs.rhodes.edu/141kirlin
- Scroll down to the last item under Resources, labeled Sept 9.
- ***Right-click*** on the four files and Save As/Save Link As each file into your new cs141 folder.

Functions

- Functions are groups of statements to which you give a name.
  - ***Defining*** a function uses the "`def`" keyword.
- That group of statements can then be referred to by that name later in the program.
  - ***Calling*** a function uses the name of the function then an opening/closing set of parentheses.



Function definitions

Function calls

```
def print_chorus():
    print("Supercali…")
     (etc)

def print_um_diddle():
    print("Um diddle diddle…")
     (etc)

def print_verse1():
    print("Because I was afraid to speak…")
     (etc)

# A function for the "main" program.
def main():
     print_chorus()            # Print the chorus
     print_um_diddle()         # Print the um diddles
     print_verse1()            # Print the 1st verse
     print_chorus()            # Print the chorus again
     print_um_diddle()         # Print the um diddles again
     print_verse2()            # Print the 2nd verse
     print_chorus()            # Print the chorus the last time

main()                         # Start the program
```

- When a function is called, Python will
  - "jump" to the first line of the function's definition,
  - run all the lines of code inside the definition, then
  - "jump" back to the point where the function was called.

- When a function is called, Python will
  - "jump" to the first line of the function's definition,
  - run all the lines of code inside the definition, then
  - "jump" back to the point where the function was called.

```
def twinkle():
  print("Twinkle twinkle little star")
  print("How I wonder what you are")

def main():
   twinkle()       # Call (run) the twinkle function.
   print("Up above the world so high")
   print("Like a diamond in the sky")
   twinkle()       # Call the twinkle function again.

main()             # Call main() to start the program.
```

Once upon a time there were three little pigs decided to build houses in the forest in which to live.

The first pig built a house out of straw.

But the Big Bad Wolf came, and he huffed, and he puffed, and he blew the house down!

The second pig built a house of sticks.

But the Big Bad Wolf came and he blew that house down too!

© www.kids-pages.com

The third pig built a house of bricks.

And the Big Bad Wolf couldn't blow down the house of bricks!

Today we're going to use functions in Python to draw the three little pigs' houses.

```
         _.-'~-_~'-._
     _.-'-_~_-~_-~-_-'-._
 _.-'_~-_~-_-~-_~_~-_~-_~'-._
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
..............................
..[]....[]....[]....[]....[]..
..............................
```

Straw

```
         _.-'~-_~'-._
     _.-'-_~_-~_-~-_-'-._
 _.-'_~-_~-_-~-_~_~-_~-_~'-._
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
||||||||||||||||||||||||||||||
||[]||||[]||||[]||||[]||||[]||
||||||||||||||||||||||||||||||
```

Sticks

```
         _.-'~-_~'-._
     _.-'-_~_-~_-~-_-'-._
 _.-'_~-_~-_-~-_~_~-_~-_~'-._
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##############################
##[]####[]####[]####[]####[]##
##############################
```

Bricks

Open threepigs-v1.py

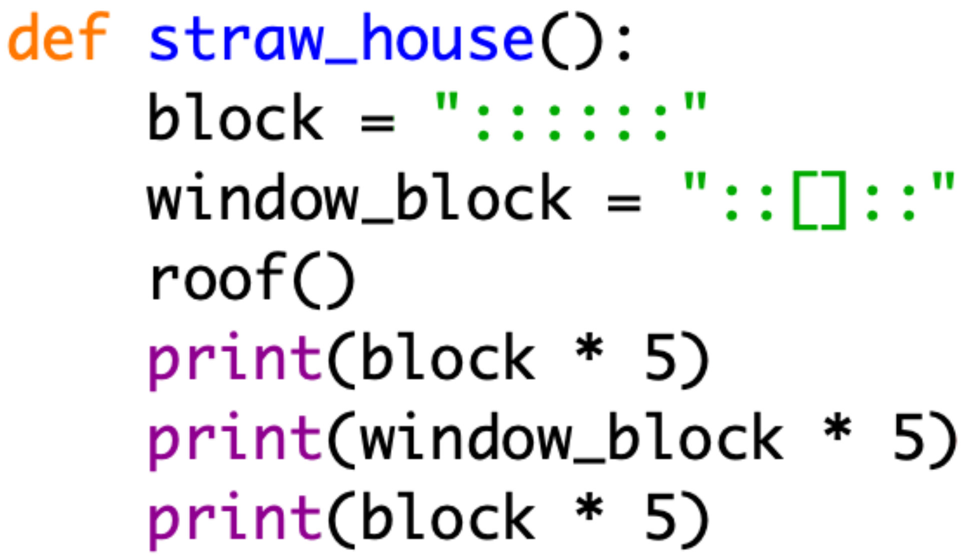
Open threepigs-v2.py

- Write a **`brick_house()`** function, similar to **`straw_house()`** and **`stick_house()`**, that draws a brick house.

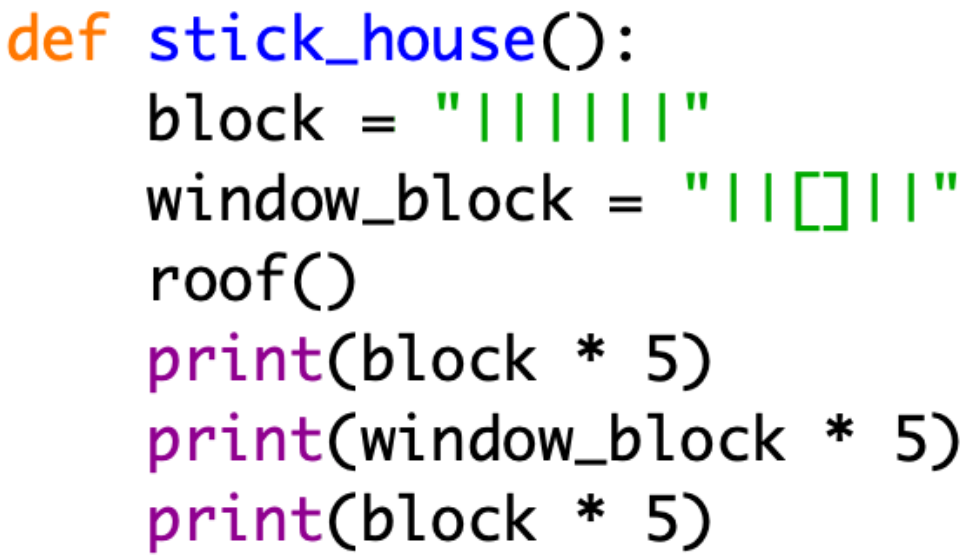
  Hint: copy and paste the stick_house definition to define brick_house.

These three functions are repetitive!

- They all draw houses, they just use different values for the block and window_block variables.



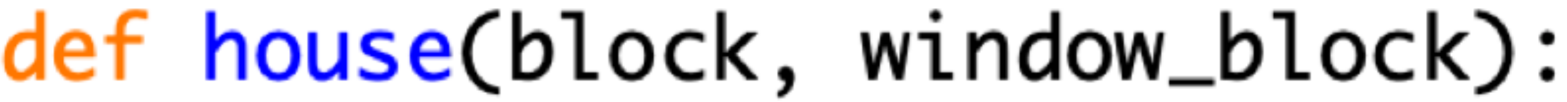
```python
def straw_house():
    block = "::::::"
    window_block = "::[]::"
    roof()
    print(block * 5)
    print(window_block * 5)
    print(block * 5)
```



```python
def stick_house():
    block = "||||||"
    window_block = "||[]||"
    roof()
    print(block * 5)
    print(window_block * 5)
    print(block * 5)
```

- We can make one function that allows us to customize the block and window_block variables.

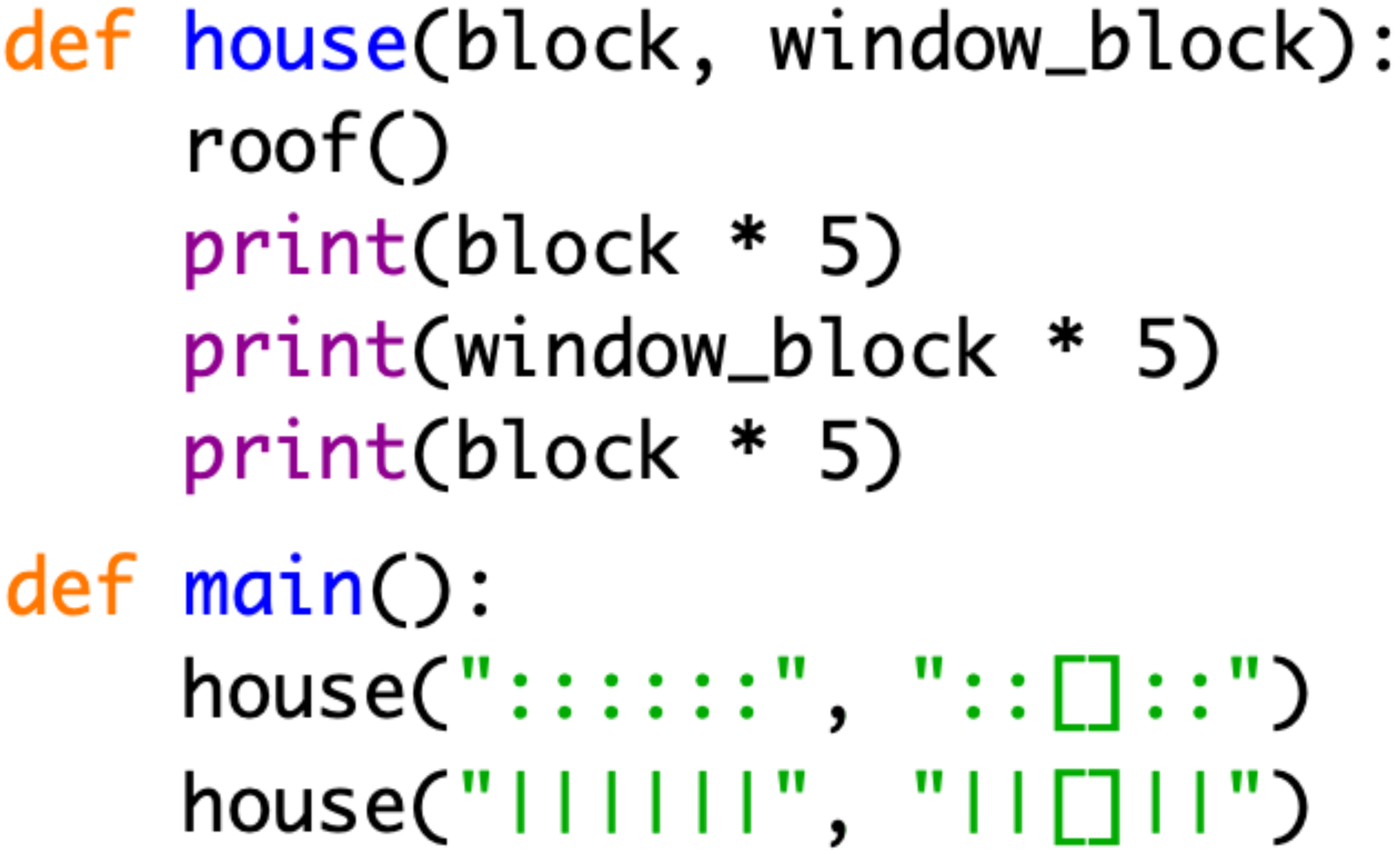
Parameters



```python
def house(block, window_block):
```

- A ***parameter*** is a variable placed in parentheses in the function definition.
- This allows the caller of the function to specify a value for that variable when the function is called.
  - This value, in the calling function, is called an ***argument***.
- As the function runs, a parameter variable can be used within the function just like any other variable.
- The point is to allow the caller of the function to send one or more pieces of information into the function that can be used to change it's behavior.

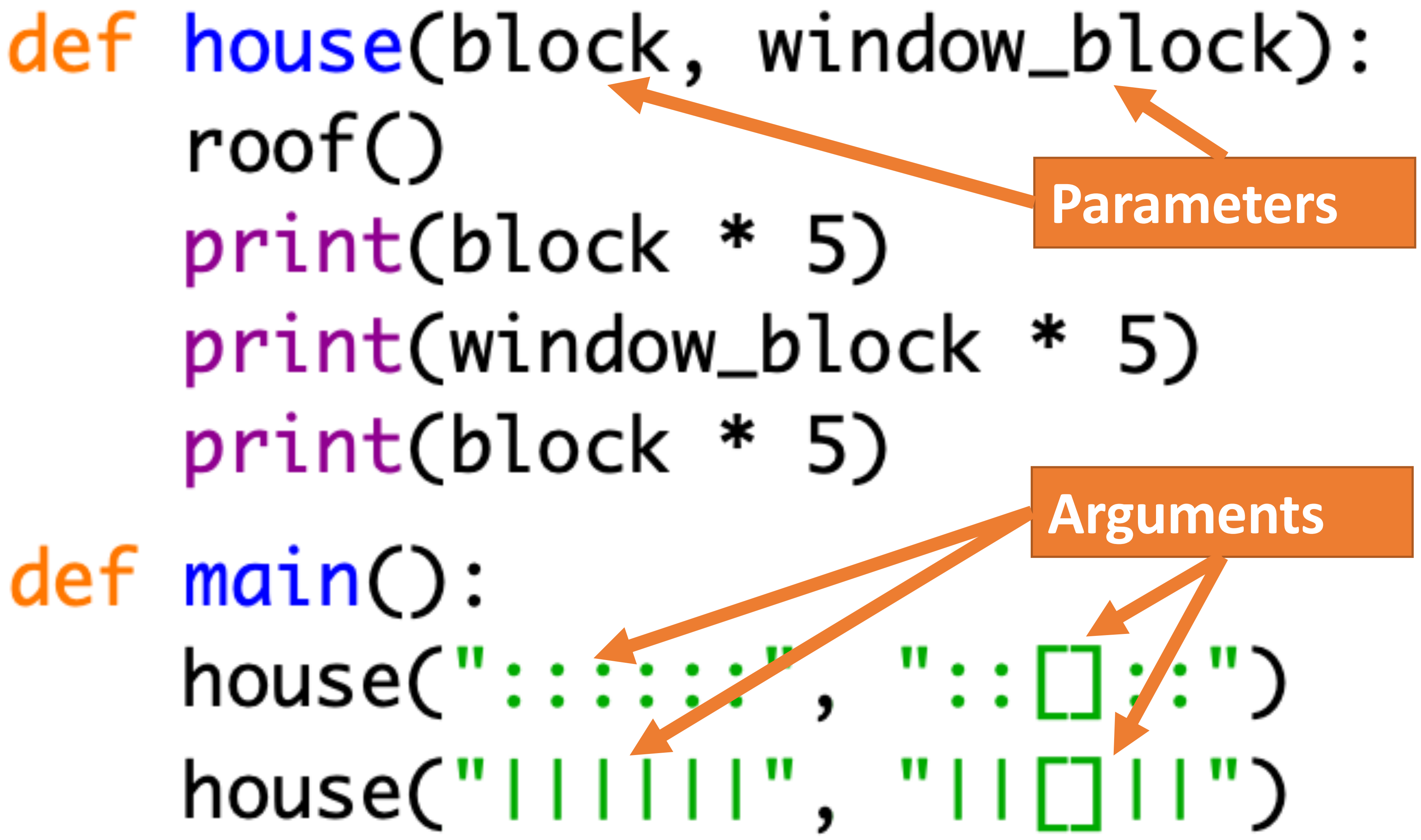
House function with **parameters**



```
def house(block, window_block):
    roof()
    print(block * 5)
    print(window_block * 5)
    print(block * 5)

def main():
    house("::::::", "::[]::")
    house("||||||", "||[]||")
```

House function with **parameters**

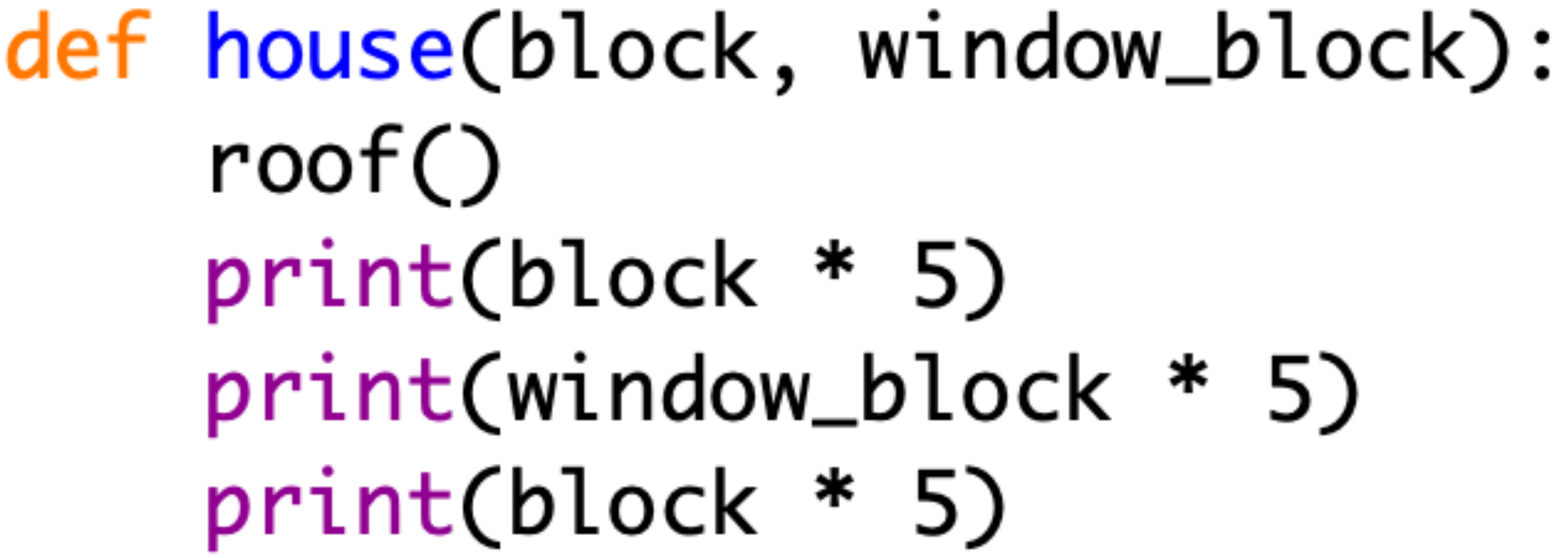


```
def house(block, window_block):
    roof()
    print(block * 5)
    print(window_block * 5)
    print(block * 5)

def main():
    house("::::::", "::□::")
    house("||||||", "||□||")
```

Parameters

Arguments

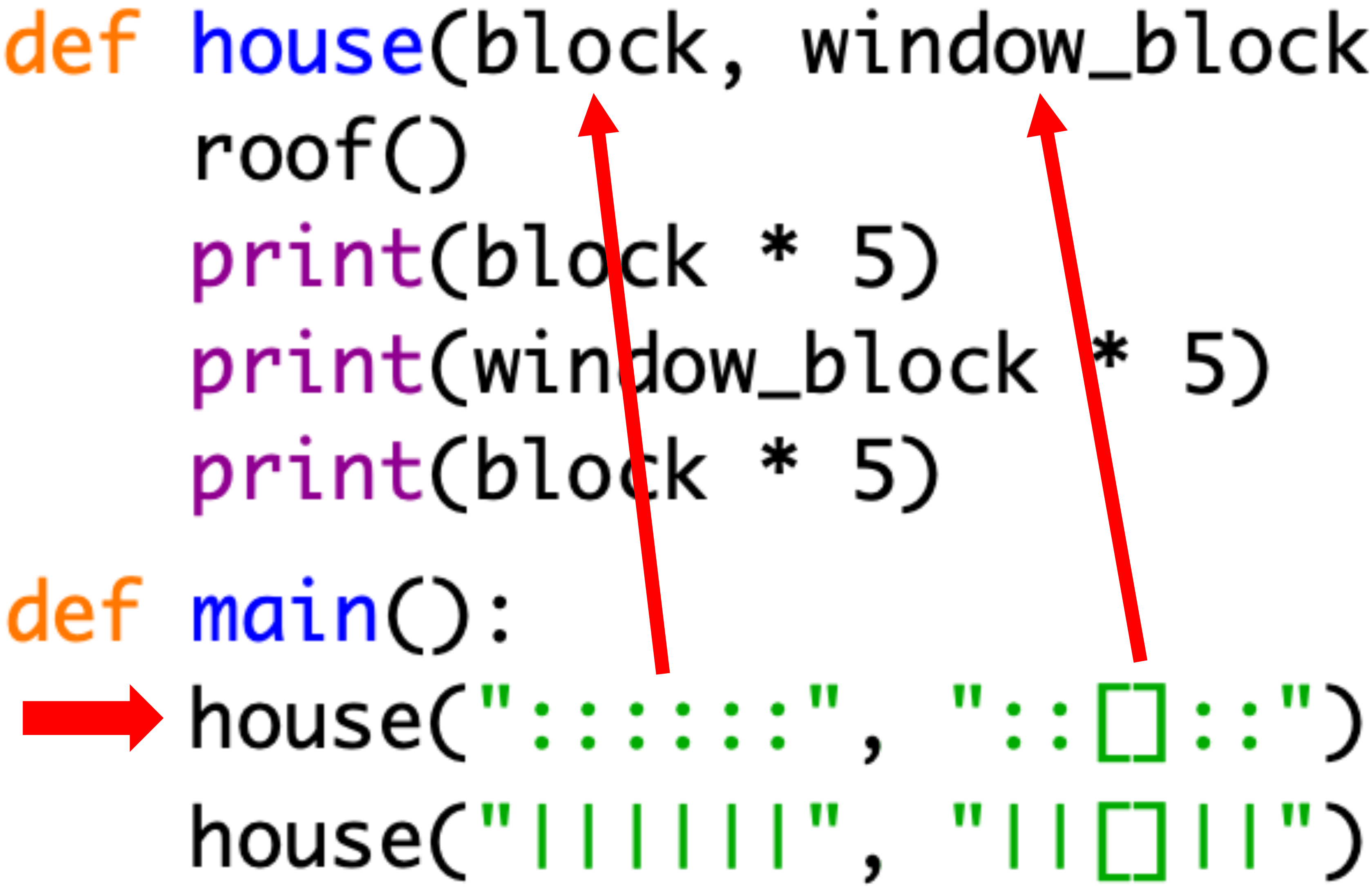


```python
def house(block, window_block):
    roof()
    print(block * 5)
    print(window_block * 5)
    print(block * 5)
```

- Notice how the parameters (block and window_block) don't have corresponding assignment statements.
- The starting value of the parameters must come from outside the function.
- When **house()** is called, the arguments inside the parentheses will be automatically assigned to the parameters before the function beings running.

Open threepigs-v3.py

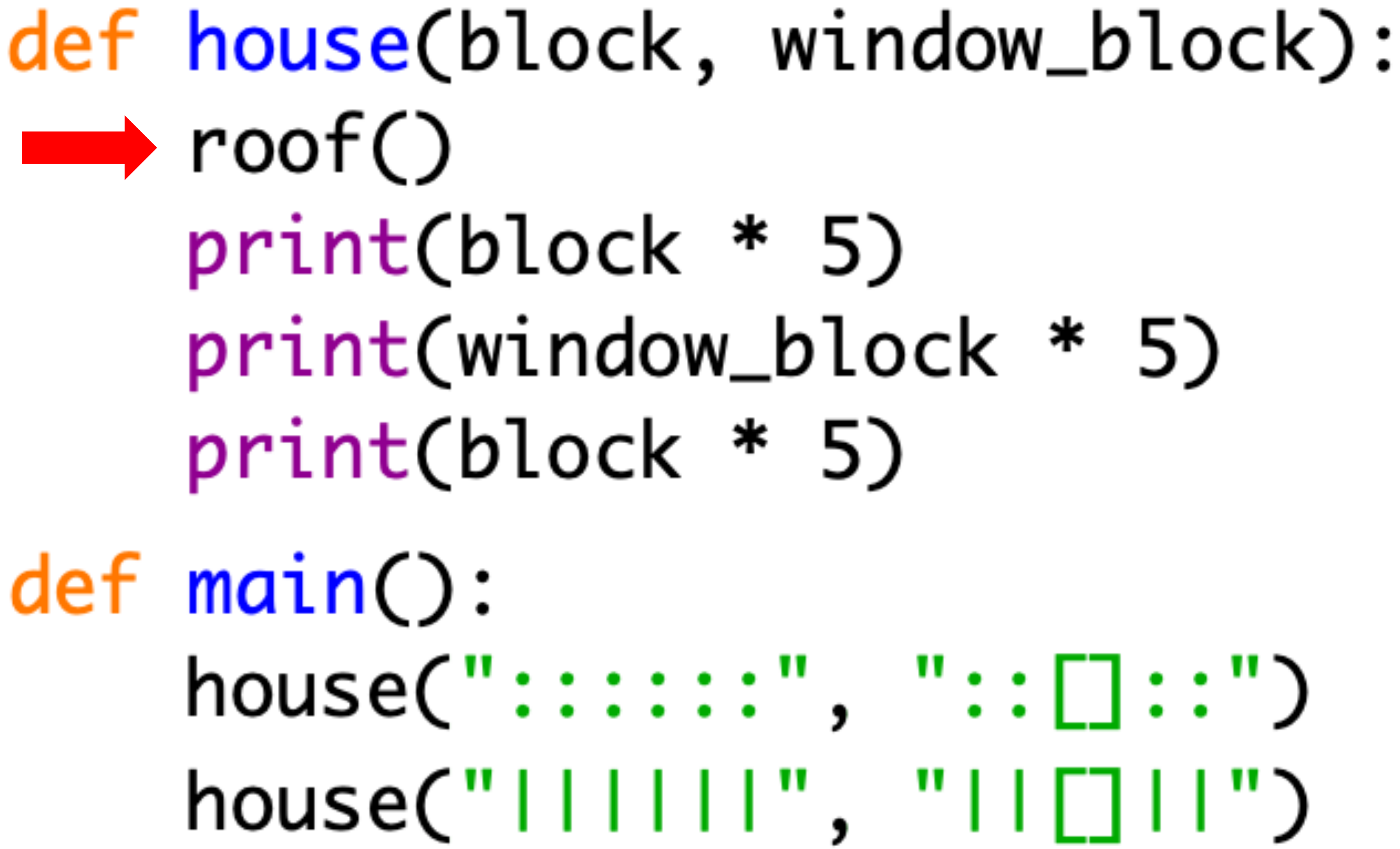
- Just run the program for now. Notice how the output is the same as version 2, but there's only one house function.



```
def house(block, window_block):
    roof()
    print(block * 5)
    print(window_block * 5)
    print(block * 5)

def main():
    house("::::::", "::□::")
    house("||||||", "||□||")
```

When Python runs the first call to house() [with the red arrow], the string arguments "::::::" and "::[]::" are copied into the parameters block and window_block.



```
def house(block, window_block):
    roof()
    print(block * 5)
    print(window_block * 5)
    print(block * 5)

def main():
    house("::::::", "::□::")
    house("||||||", "||□||")
```

Then Python begins running the lines of code inside the house function, one by one.

```
def house(block, window_block):
    roof()
    print(block * 5)
    print(window_block * 5)
    print(block * 5)

def main():
    house("::::::", "::□::")
    house("||||||", "||□||")
```

"::::::" used for block

Then Python begins running the lines of code inside the house function, one by one.

```
def house(block, window_block):
    roof()
    print(block * 5)
    print(window_block * 5)
    print(block * 5)

def main():
    house("::::::", "::[]::")
    house("||||||", "||[]||")
```

"::::::" used for window_block

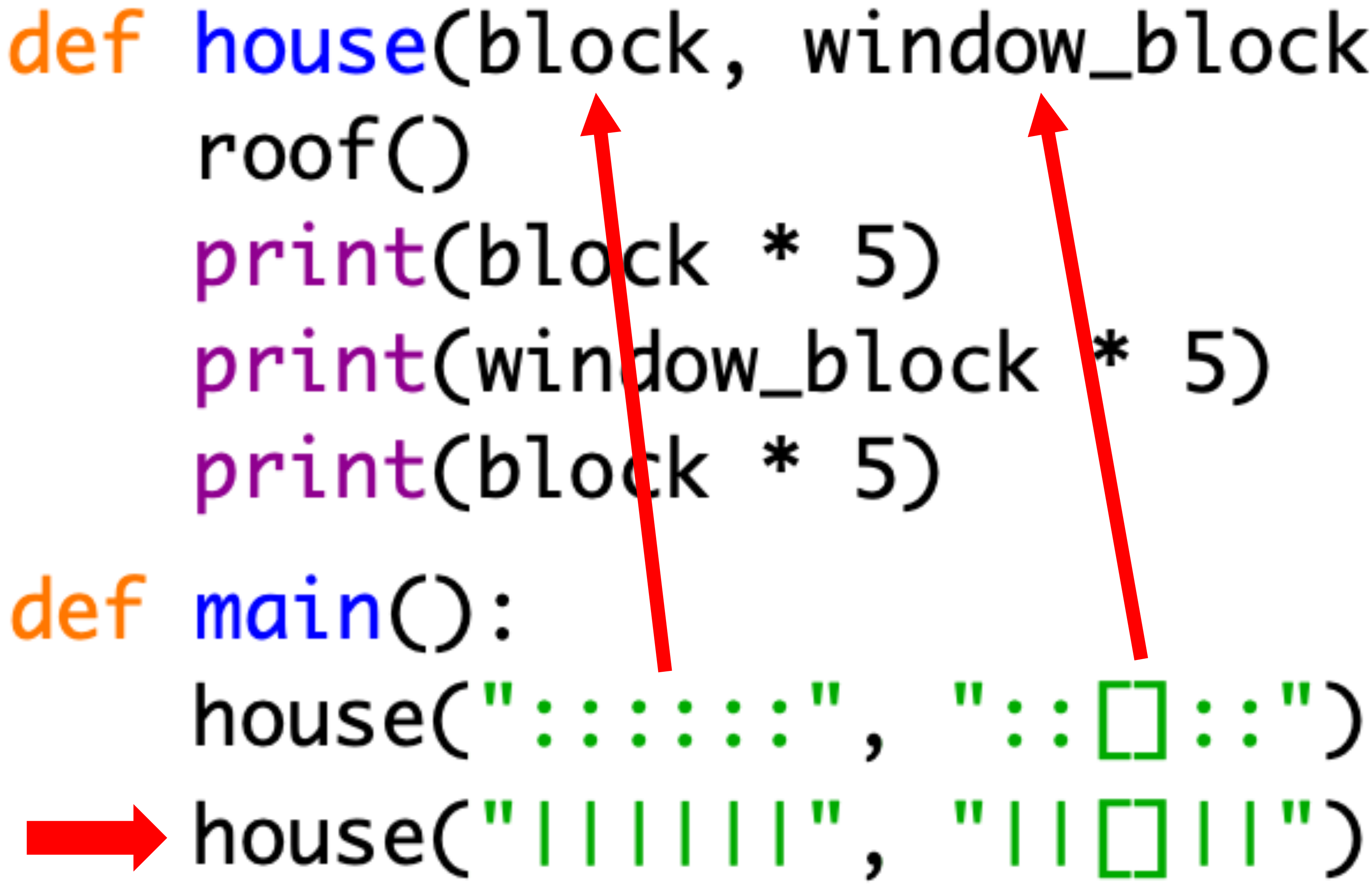
Then Python begins running the lines of code inside the house function, one by one.

```
def house(block, window_block):
    roof()
    print(block * 5)
    print(window_block * 5)
➡   print(block * 5)

def main():
    house("::::::", "::□::")
    house("||||||", "||□||")
```

"::::::" used for block

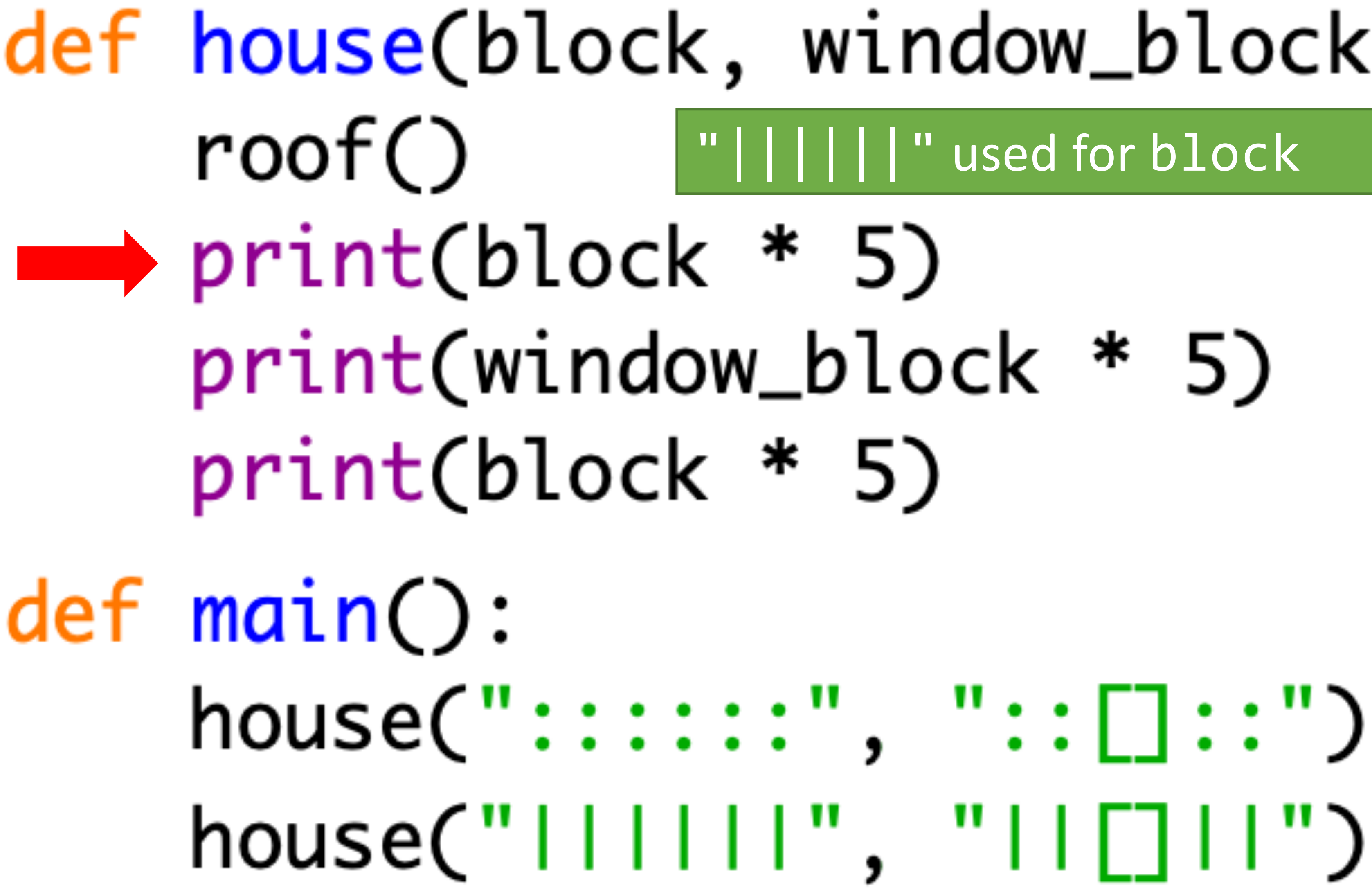
Then Python begins running the lines of code inside the house function, one by one.



```
def house(block, window_block):
    roof()
    print(block * 5)
    print(window_block * 5)
    print(block * 5)

def main():
    house("::::::", "::[]::")
    house("||||||", "||[]||")
```

When Python runs the second call to house() [with the red arrow], the string arguments "::::::" and "::[]::" are copied into the parameters block and window_block.

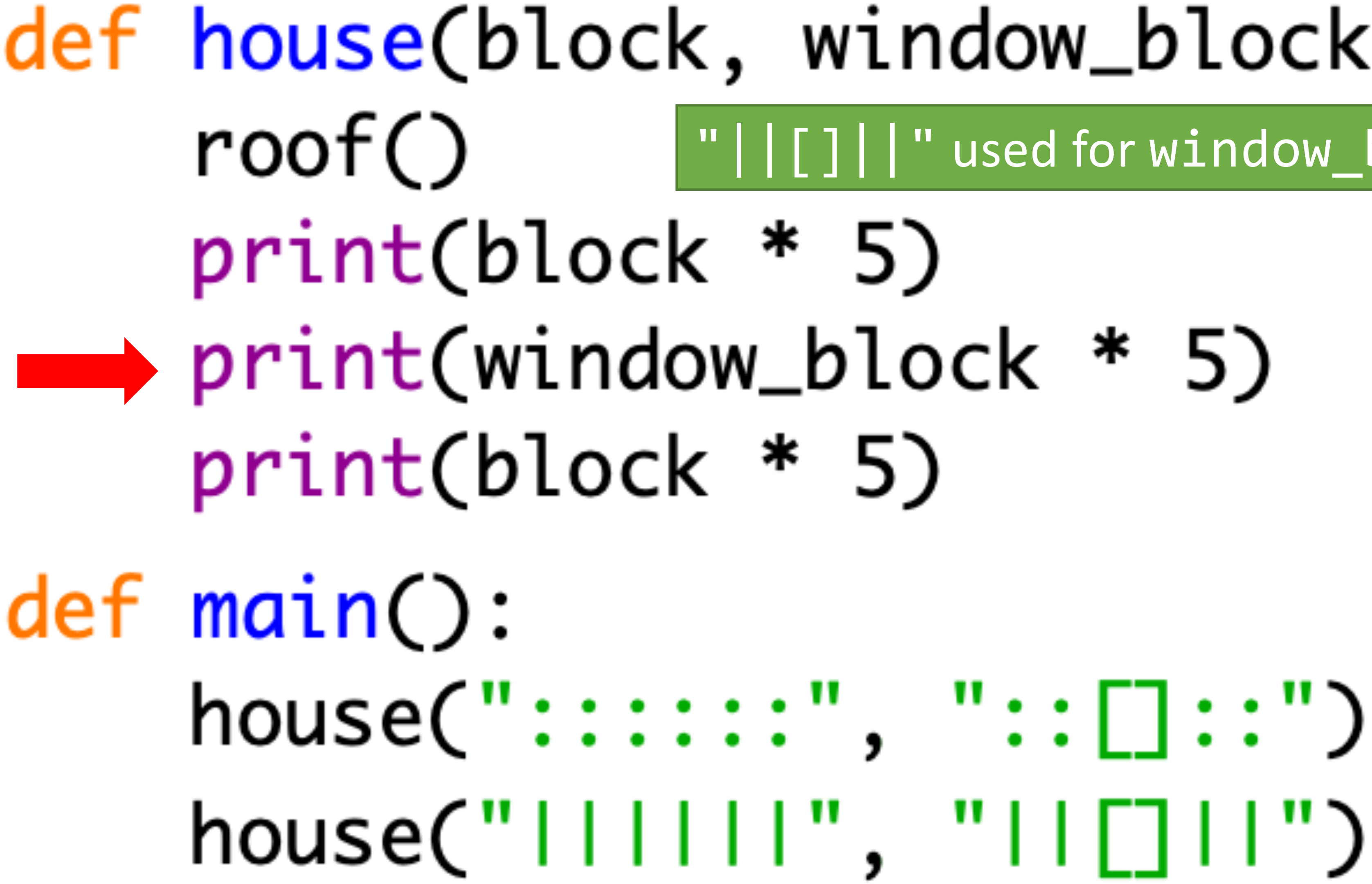


```
def house(block, window_block):
    roof()
    print(block * 5)
    print(window_block * 5)
    print(block * 5)

def main():
    house("::::::", "::□::")
    house("||||||", "||□||")
```

"||||||" used for block

Then Python begins running the lines of code inside the house function, one by one.

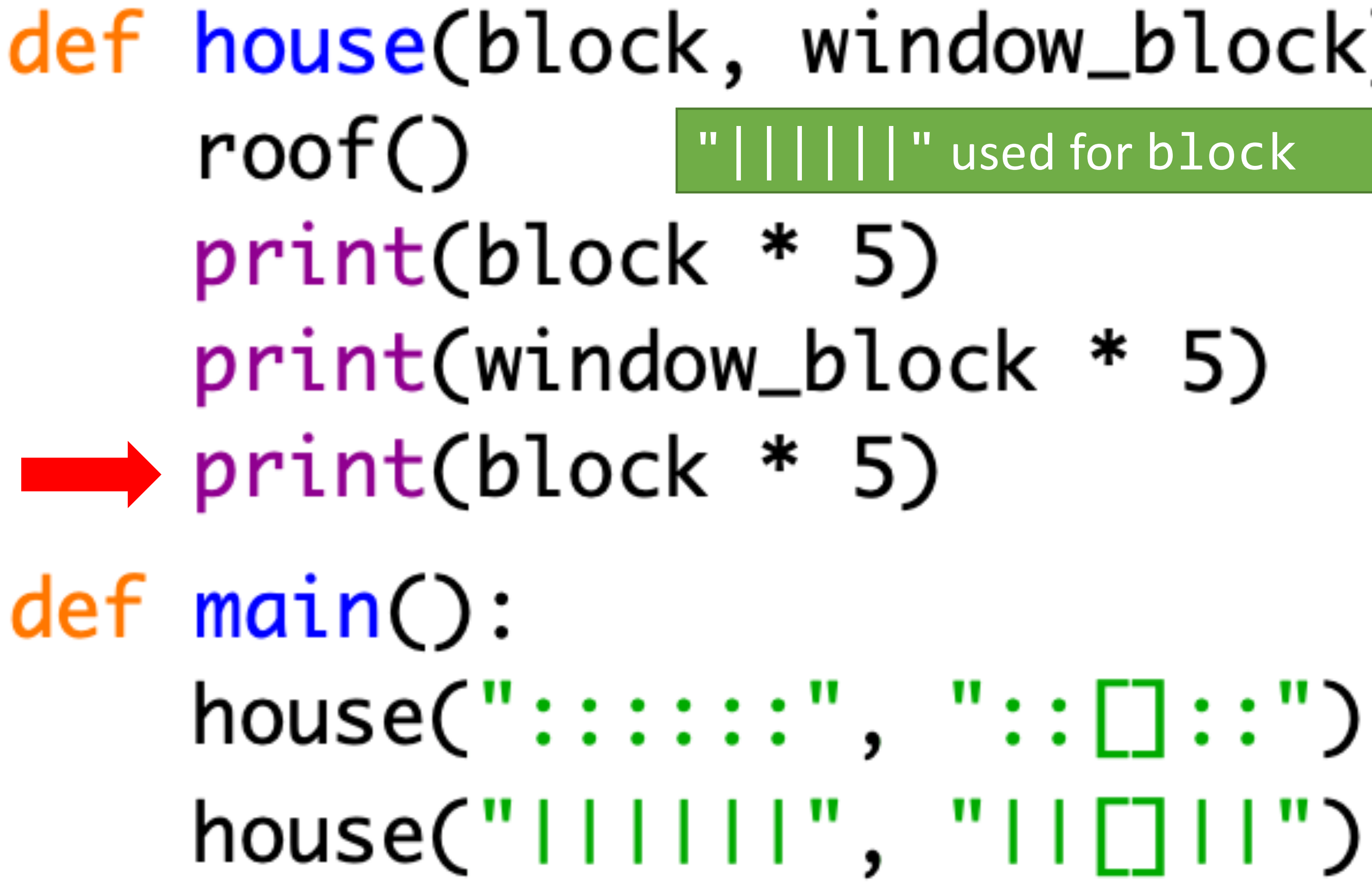


```
def house(block, window_block):
    roof()
    print(block * 5)
    print(window_block * 5)
    print(block * 5)

def main():
    house("::::::", "::[]::")
    house("||||||", "||[]||")
```

"||[]||" used for window_block

Then Python begins running the lines of code inside the house function, one by one.



```
def house(block, window_block):
    roof()
    print(block * 5)
    print(window_block * 5)
    print(block * 5)

def main():
    house("::::::", "::□::")
    house("||||||", "||□||")
```

"||||||" used for block

Then Python begins running the lines of code inside the house function, one by one.

Open threepigs-v3.py

- Go to the **`main()`** function definition, and add a third call to the **`house()`** function with appropriate arguments so that your program prints the brick house when the program is run (in addition to the straw & stick houses).

You've seen arguments already

- `name = input("What is your name? ")`
- `x = 5`
- `y = 2`
- `print("x is", x, "y is", y)`
- `print("their sum is", x + y)`

Arguments can be variables, literals, or math expressions.

(Anything you could put on the right side of a variable assignment statement can be an argument.)

Continuing in threepigs-v3.py

- Let's let the user of our program customize their own house picture while the program is running!
- Go to the **`main()`** function definition, and after your third call to house (the one you just wrote), add ***two (string) input statements***.
- The ***first*** input statement should ask the user for a 6-symbol string that will be used as the basic house block.
- The ***second*** input statement should ask the user for a 6-symbol string to be used as the window block.
- Then call **`house()`**, using the variable values the user typed in to display a house of the user's choice.

Local variables

- Any variable used as a parameter inside a function is "owned" by that function, and is ***invisible*** to all other functions.
- These are called ***local variables*** because they can only be used "locally" (within their own function).
- Any variable created inside a function is also a local variable and cannot be seen outside of that function.

```
def some_function(x):
    print(“Inside the function, x is”, x)
    x = 17
    print(“Inside the function, x is changed to”, x)

def main():
    x = 2
    print(“Before the function call, x is”, x)
    some_function(x)
    print(“After the function call, x is”, x)


main()
```

Output:

```
Before the function call, x is 2
Inside the function, x is 2
Inside the function, x is 17
After the function call, x is 2
```

Wait --- what?

- There is no permanent connection between the `x` in `main` and the `x` in `some_function`.
- Arguments are passed --- one way only --- from `main` to `some_function` when `main` calls `some_function`.
  - This copies main's value of `x` into `some_function`'s `x`.
- Any assignments to `x` inside of `some_function` do not come back to `main` .

Continuing in threepigs-v3.py

- Let's add a third parameter to **`house()`** that lets the caller of this function control the number of floors the house has!
- Inside the **`house()`** function, add a third parameter called `floors`. This will be an integer variable.
- Change the code inside the function definition so that if `floors` is 1, the house looks like it normally does. But if `floors` is 2, it displays the house with a 2nd floor.
- You will need to change all of your calls to **`house()`** to reflect that it requires 3 arguments now!
  - Make the straw house have 1 floor, but the stick & brick houses have 2 floors. For the user-designed house, add an input statement to let the user choose the number of floors.
- Challenge: Change your function to allow 3 or 4 floors!

```
         _.-'~-_~'-._
     _.-'-_~_-~_-~-_-'-._
 _.-'_~-_~-_-~-_~_~-_~-_~'-._
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
::::::::::::::::::::::::::::::
::[]::::[]::::[]::::[]::::[]::
::::::::::::::::::::::::::::::
```

```
         _.-'~-_~'-._
     _.-'-_~_-~_-~-_-'-._
 _.-'_~-_~-_-~-_~_~-_~-_~'-._
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
||||||||||||||||||||||||||||||
||[]||||[]||||[]||||[]||||[]||
||||||||||||||||||||||||||||||
||[]||||[]||||[]||||[]||||[]||
||||||||||||||||||||||||||||||
```

```
         _.-'~-_~'-._
     _.-'-_~_-~_-~-_-'-._
 _.-'_~-_~-_-~-_~_~-_~-_~'-._
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##############################
##[]####[]####[]####[]####[]##
##############################
##[]####[]####[]####[]####[]##
##############################
```

```
         _.-'~-_~'-._
     _.-'-_~_-~_-~-_-'-._
 _.-'_~-_~-_-~-_~_~-_~-_~'-._
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
///\\\///\\\///\\\///\\\///\\\
//[]\\//[]\\//[]\\//[]\\//[]\\
///\\\///\\\///\\\///\\\///\\\
//[]\\//[]\\//[]\\//[]\\//[]\\
///\\\///\\\///\\\///\\\///\\\
```