



Python Quick Tips

Defining Functions



Python Quick Tips #5

Defining Functions

Built-In



So far, most functions
we've used are **built-in**

Defined



But we can actually
define **our own!**

Why?



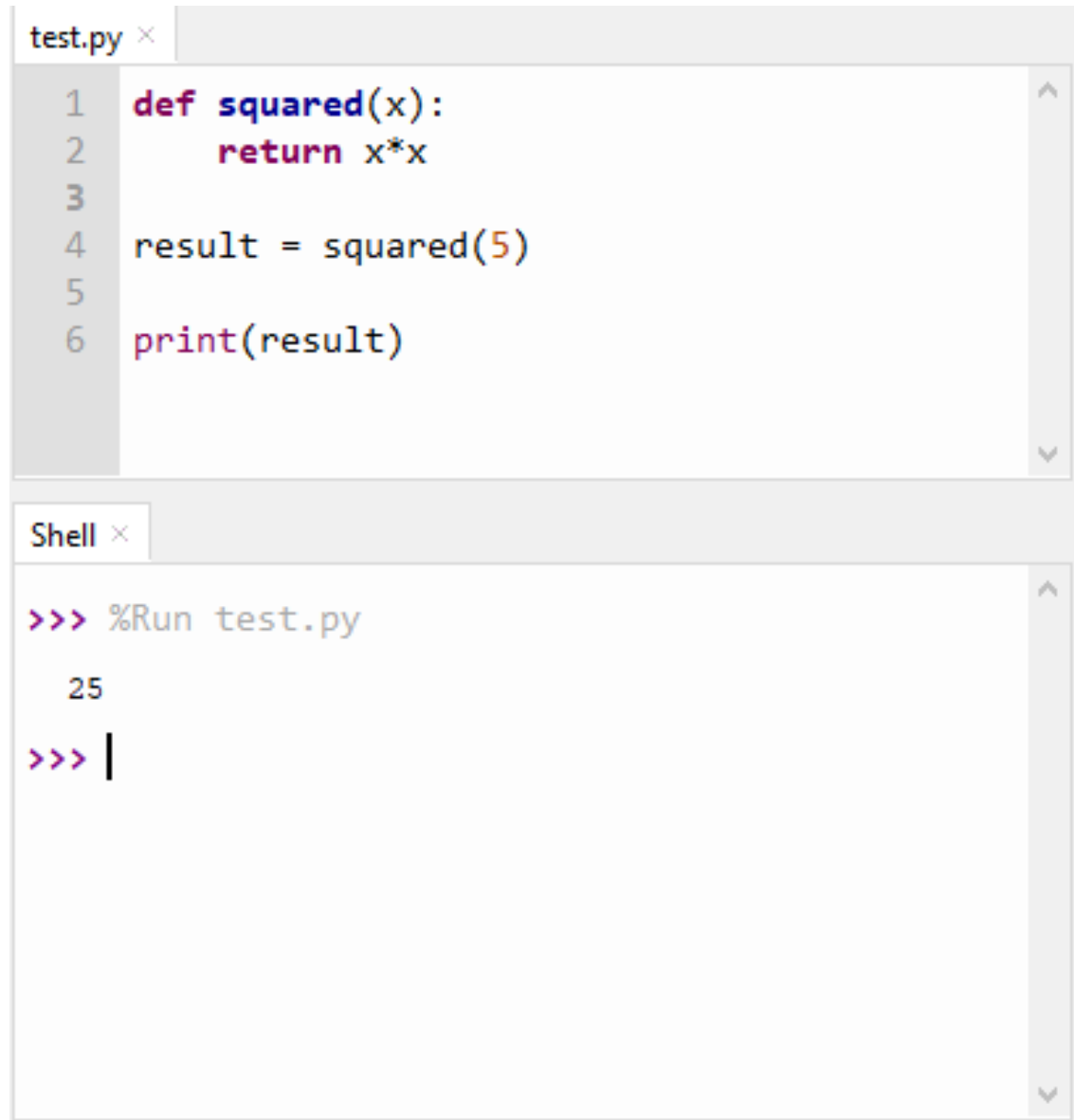
Simplify our script
Consolidate **repetition**

Define (def)

Syntax

```
def name(var):  
  (tab)code  
  (tab)etc.  
return = result
```

Define (def)

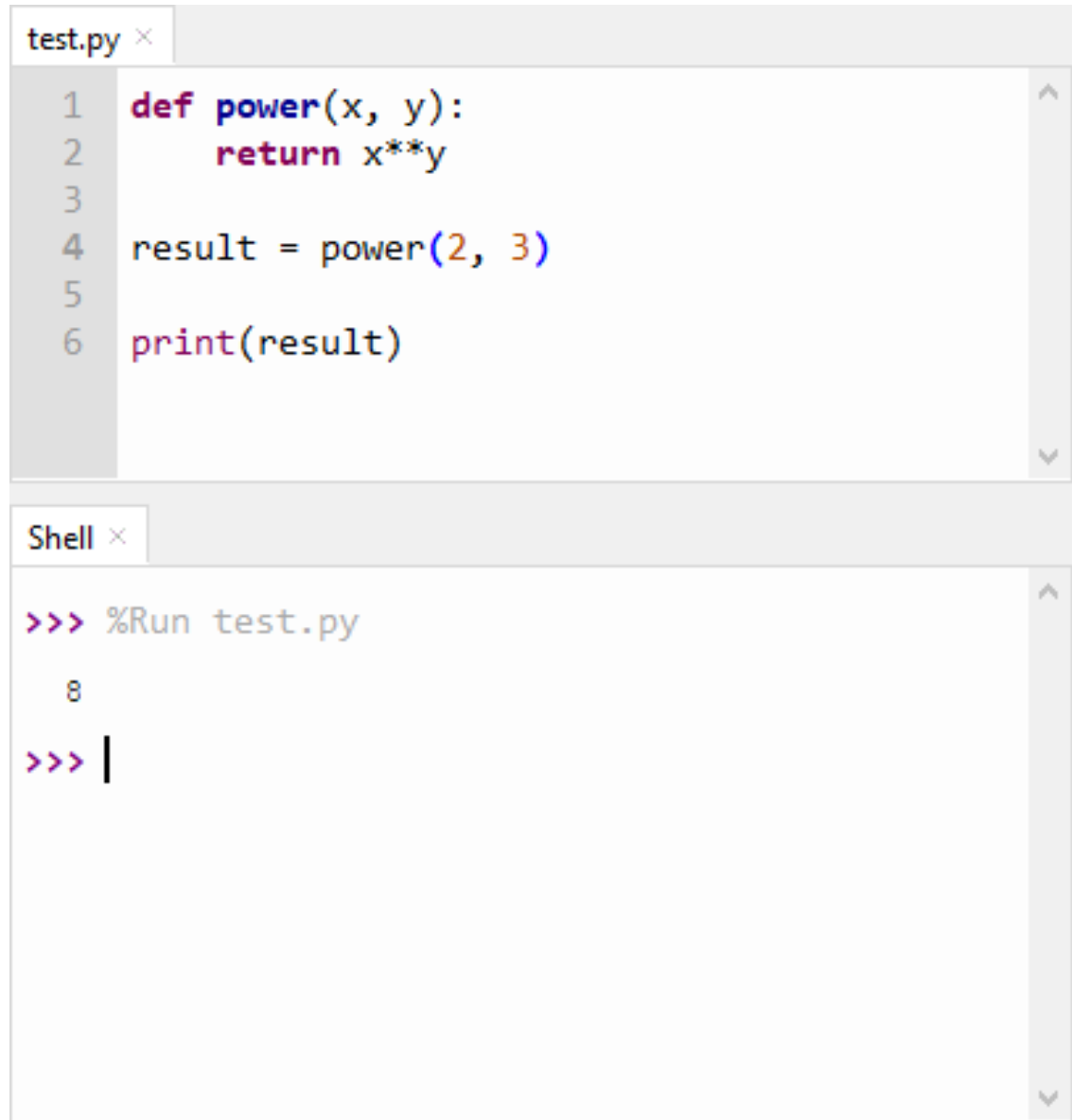


The image shows a screenshot of a Python IDE interface. The top pane, titled 'test.py', contains a Python script with six lines of code. The first two lines define a function named 'squared' that takes a parameter 'x' and returns 'x*x'. The next three lines call this function with the argument '5', store the result in a variable named 'result', and then print the value of 'result'. The bottom pane, titled 'Shell', shows the output of running the script. It displays the command prompt '>>> %Run test.py', followed by the output '25', and then a new prompt '>>> |' with a vertical cursor.

```
test.py ×
1  def squared(x):
2      return x*x
3
4  result = squared(5)
5
6  print(result)

Shell ×
>>> %Run test.py
25
>>> |
```

Multiple Variables



The image shows a screenshot of a Python IDE with two panels. The top panel, titled 'test.py', contains a Python script with six lines of code. The bottom panel, titled 'Shell', shows the execution of the script, resulting in the output '8'.

```
test.py ×  
1 def power(x, y):  
2     return x**y  
3  
4 result = power(2, 3)  
5  
6 print(result)  
  
Shell ×  
>>> %Run test.py  
8  
>>> |
```


Default Values

test.py ×

```
1 def power(x, y=2):  
2     return x**y  
3  
4 result = power(2)  
5  
6 print(result)
```

Shell ×

```
>>> %Run test.py  
4  
>>> |
```

Local Variables

```
test.py ×
1  def power(x, y=2):
2      return x**y
3
4  print(y)

Shell ×
>>> %Run test.py
Traceback (most recent call last):
  File "C:\Users\Gavin\Desktop\test.py", line 4,
in <module>
    print(y)
NameError: name 'y' is not defined
>>> |
```

LEGB

```
test.py * x
1  def enclosed(x)
2      e_enclosing = 'enclosing'
3
4      def local(y)
5          l_local = 'local'
6
7  g_global = 1
8
9  b_builtin = range(1,5,1)
```

Enclosing

Local

Another Example

test.py ×

```
1 def divisible(number, divisor):
2     rem = number%divisor
3     return rem == 0
4
5 check1 = divisible(15,4)
6 check2 = divisible(15,3)
7
8 print([check1, check2])
```

Shell ×

```
>>> %Run test.py
[False, True]
>>> |
```



Next on #6
If Statements



Python Quick Tips

Defining Functions