Functions

Defining a Function

```
def add(a, b):
    """
    adds two numbers
    """
    z = a + b
    return z
```

- general syntax:
  (1) def statement (name and parameters)
  (2) docstring
  (3) statements to compute
  (4) (optional) return statement

```
>>> x = add(10, 15)

>>> x

15
```

Functions as First Class Objects

- functions can be passed as arguments
- can be used as any other value

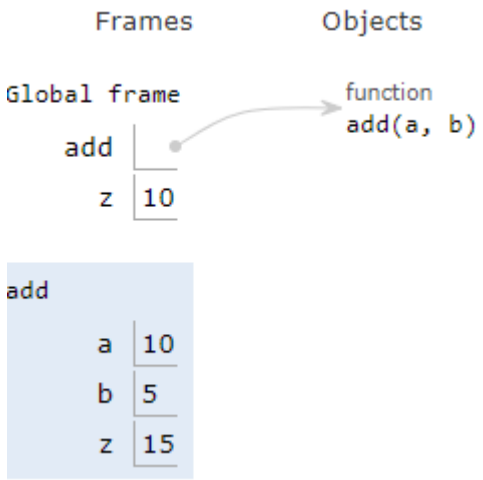
```
def add(a, b):
    return a + b
```

```
>>> print(add)
<function add at 0x7f1566d8cde8>
>>> print(add(3,5))
8
>>> print(type(add))
<type 'function'>
```

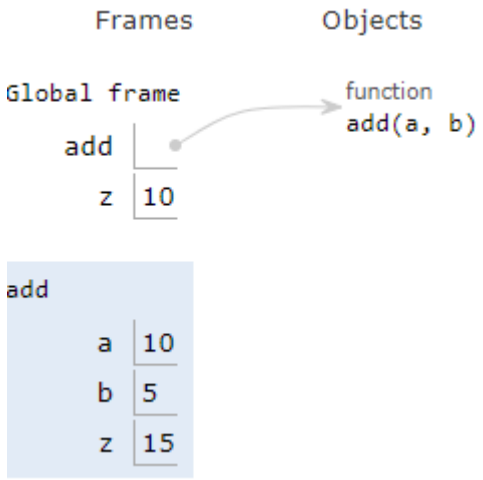
Function and Variable Scope

```
def add(a, b):
    z = a + b
    return z
```

```
>>> z = 10;  z = add(z, 5)
```




```
>>> z
```

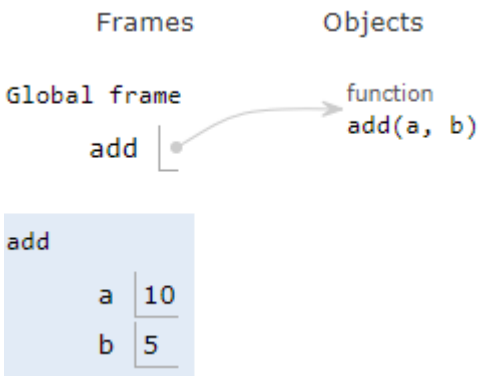


Functions With Positional Parameters

```
def add(a, b):
    z = a + b
    return z
```

```
>>> res = add(10, 5)
```

- parameters are bound by position

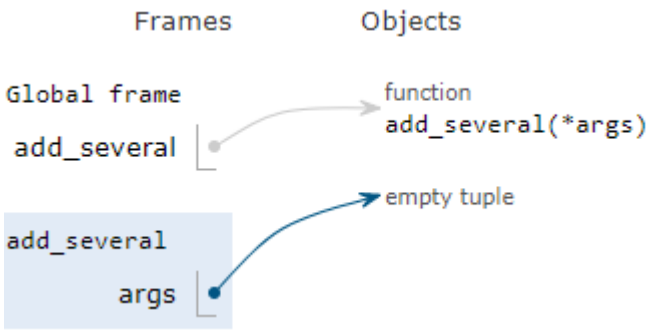



```
>>> res
15
```

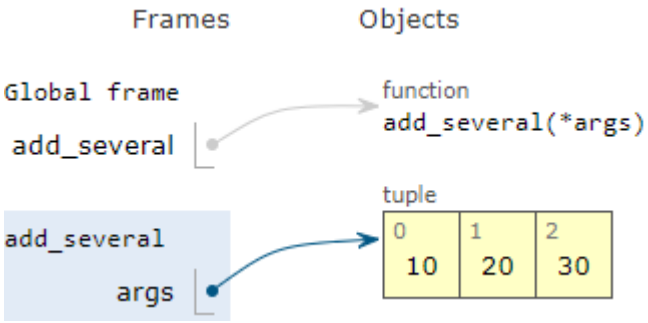
Variable Positional Args

```
def add_several(*args):
    result = 0
    for x in args:
        result = result + x
    return result
```

```
>>> add_several()
```



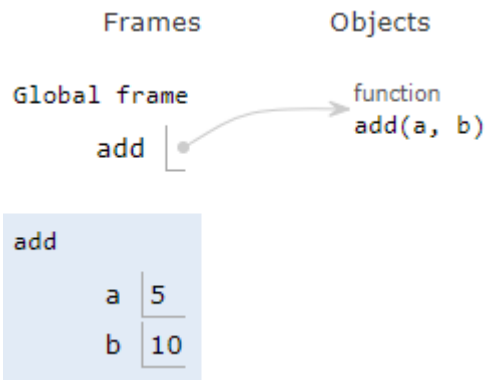

```
>>> add_several(10, 20, 40)
```



Optional Parameters

```
def add(a, b = 10):
    z = a + b
    return z
```

```
>>> add(5)
15
```



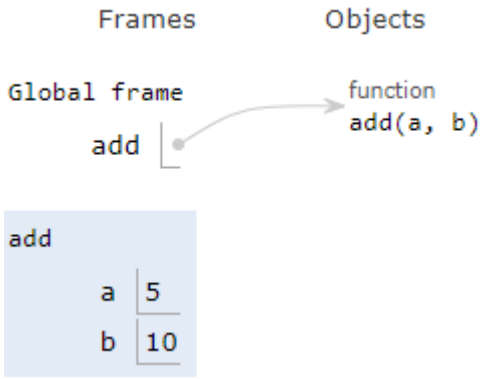
- default is bound to the function upon creation (not when it is called)

Keyword Arguments

```
def add(a, b):
    z = a + b
    return z
```

```
>>> res = add(b=10, a=5)
```

- parameters are bound by name




```
>>> res
15
```

Parameter Passing

- **call by value**:
  1. function copies values of parameters
  2. any changes to parameters are not visible to calling function

- **call by reference**:
  1. function uses address of parameters
  2. any changes to parameters are visible

- Python uses **call by assignment**

Call by Value Parameter Passing

- call by value:
  1. function copies values of parameters
  2. any changes to parameters are not visible to calling function

```
def modify(data):
    data.append('Mozzarella')
```

```
>>> x = [1, 2, 3]
>>> modify(x)
>>> print('received x = ', x)
```

- call by value should return

```
received x = [1, 2, 3']
```

Call by Reference Parameter Passing

- call by reference:
  1. function uses address of parameters
  2. any changes to parameters are visible

```
def modify(data):
    data = 'Brie’
```

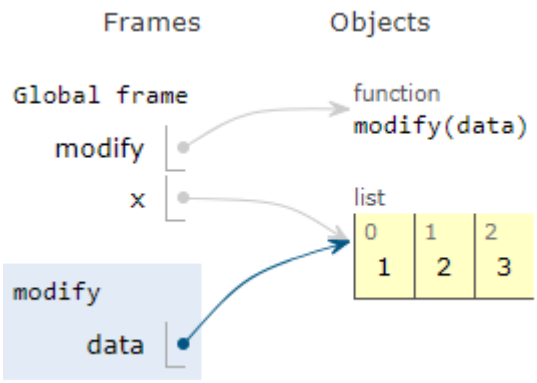
```
>>> x = 'Mozzarella’
>>> modify(x)
>>> print('received x = ', x)
```

- call by reference should return:

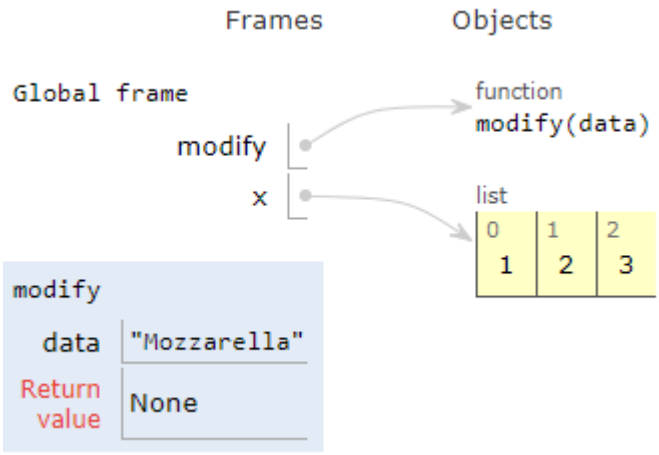
```
received x = 'Brie’
```

Mutable Objects with Rebinding (in detail)

```
>>> x = [1, 2, 3]
```




```
>>> modify(x)
```




```
>>> print('received x = ', x)
received x = [ 1, 2, 3]
```

Parameter Passing: Mutable Objects without Rebinding

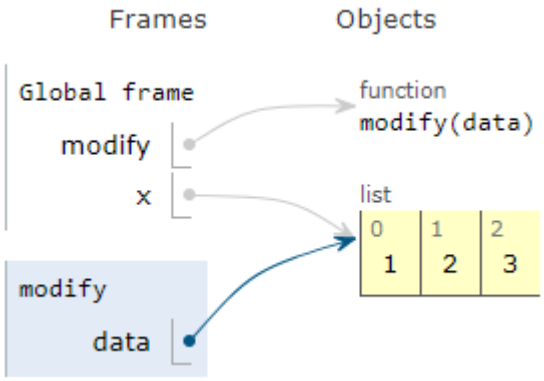
- called function gets a reference to the object and can change it
- calling function points to mutated object without rebinding − changes are visible

```
def modify(data):
    data.append('Mozzarella')
```

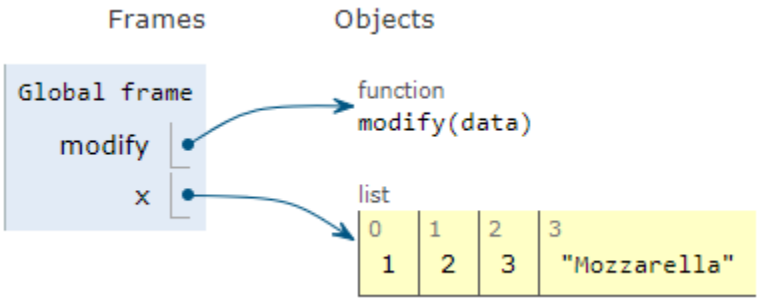
```
>>> x = [1, 2, 3]
>>> modify(x)
>>> print('received x = ', x)
received x = [1, 2, 3, 'Mozzarella']
```

Mutable Objects without Rebinding (details)

- data points to the same object



- modify object




```
>>> print('received x = ', x)
received x = [1, 2, 3, 'Mozzarella']
```

Passing Immutable Objects as Parameters

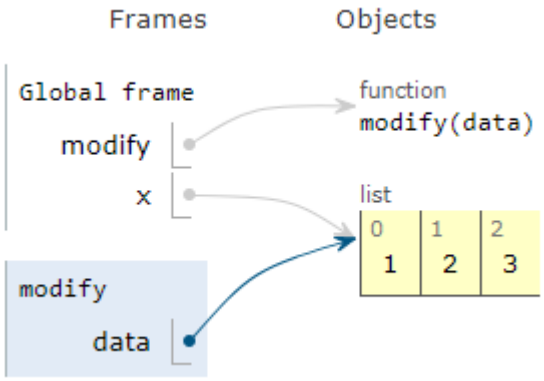
- called function gets a reference to the object and cannot change it
- calling function points to same object: no changes to the object are possible

```
def modify(data):
    data = 'Brie’
```

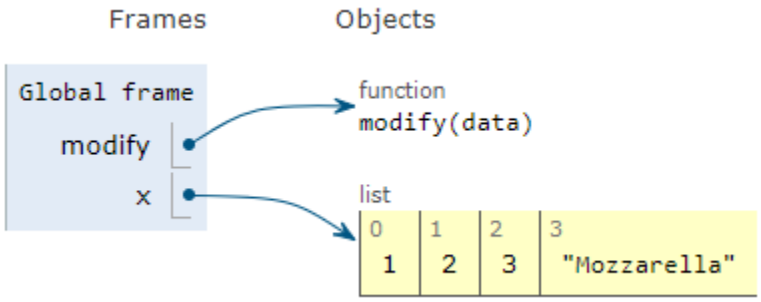
```
>>> x = 'Mozzarella’
>>> modify(x)
>>> print('received x = ', x)
received x = 'Mozzarella'
```

Mutable Objects without Rebinding (details)

- data points to the same object



- modify object




```
>>> print('received x = ', x)
received x = [1, 2, 3, 'Mozzarella']
```

Parameter Passing: Mutable Objects with Rebinding

- called function gets a reference to the object and can change it
- calling function points to mutated object with rebinding − changes are invisible

```
def modify(data):
    data = ['Mozzarella']
```

```
>>> x = [1, 2, 3]
>>> modify(x)
>>> print('received x = ', x)
received x = [1, 2, 3, 'Mozzarella']
```

Parameter Passing Summary

- arguments in functions are objects
- all variables reference an Object
- arguments are passed by reference
- mutable arguments can be changed in a function – changes are visible outside
- changes to immutable arguments result in new object(s) creation – changes are not visible outside

yield Statement

- suspends function execution
- produces a sequence of values to caller
- retains enough state to resume
- use yield to iterate over a sequence
- if function body contains yield, function becomes a generator

yield Example

```
def compute_next_cube():
    x = 1
    while True:
        yield x**3
        x = x + 1
```

- code to test

```
for result in compute_next_cube():
    if result > 1000:
        break
    print result
```

- prints

1, 8, 27 ,64, 125, 216, 343, 512, 729, 1000

Review Problems

Interview Problem

- why are functions considered first class objects in Python?

Interview Problem

- how do you create a Python function?

Interview Problem

- how does a function return values?

Interview Problem

- what happens when a function does not have a return statement?
- is this valid?

Interview Problem

- what will be the output of the following?

```
def func(x,*y,**z):
    print z
```

```
>>> func(1,2,3)
```

Interview Problem

- what does this code output?

```
def f(x, l=[]):
    for i in range(x):
        l.append(i*i)
print(l)
```

Interview Problem

- what does *args and **kwargs mean?

Interview Problem

- are function parameters passed by value or by reference?

Interview Problem

- differentiate local and global variables

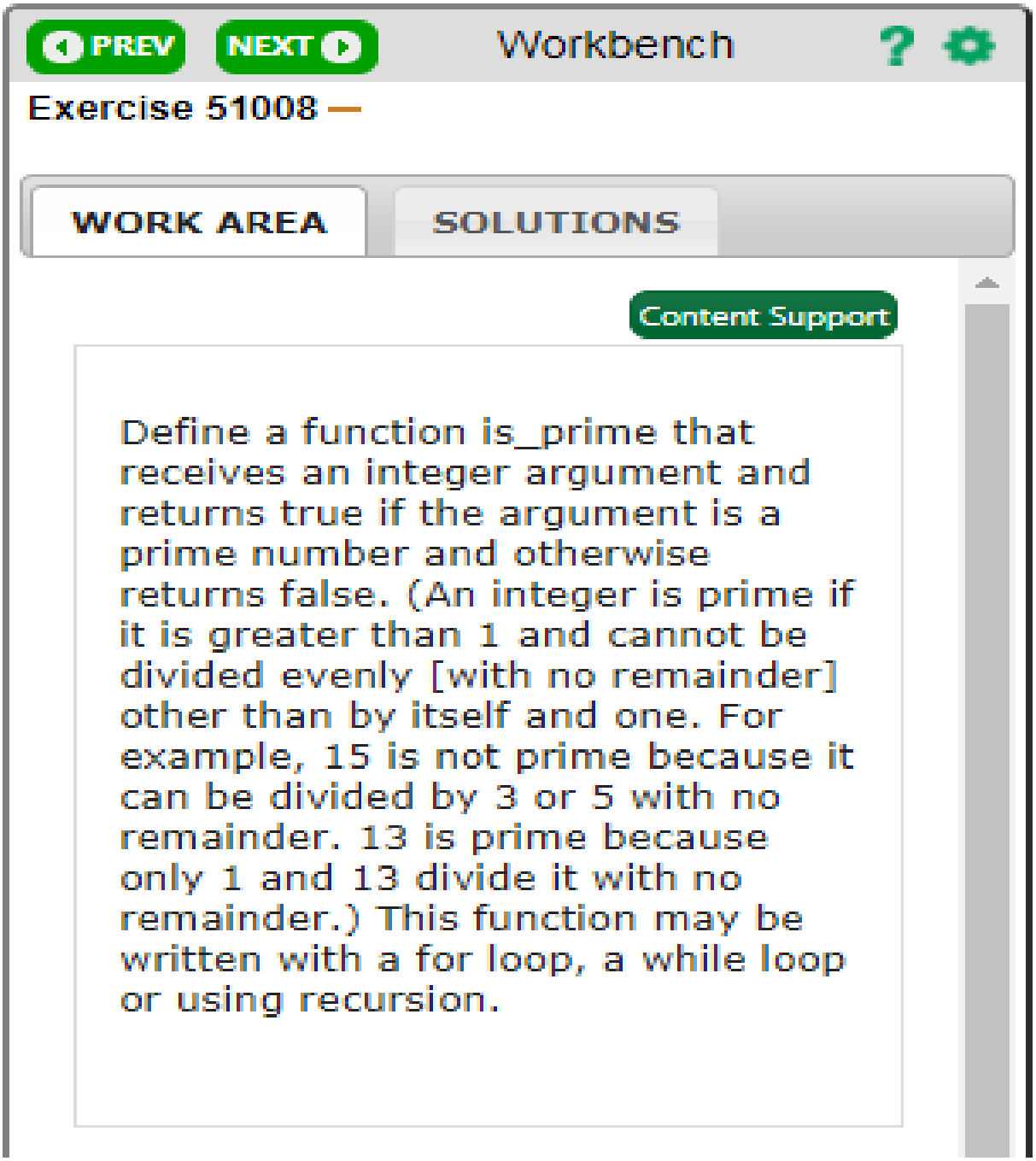
Interview Problem

- what are decorators?

Interview Problem

- what does the *yield* keyword do?

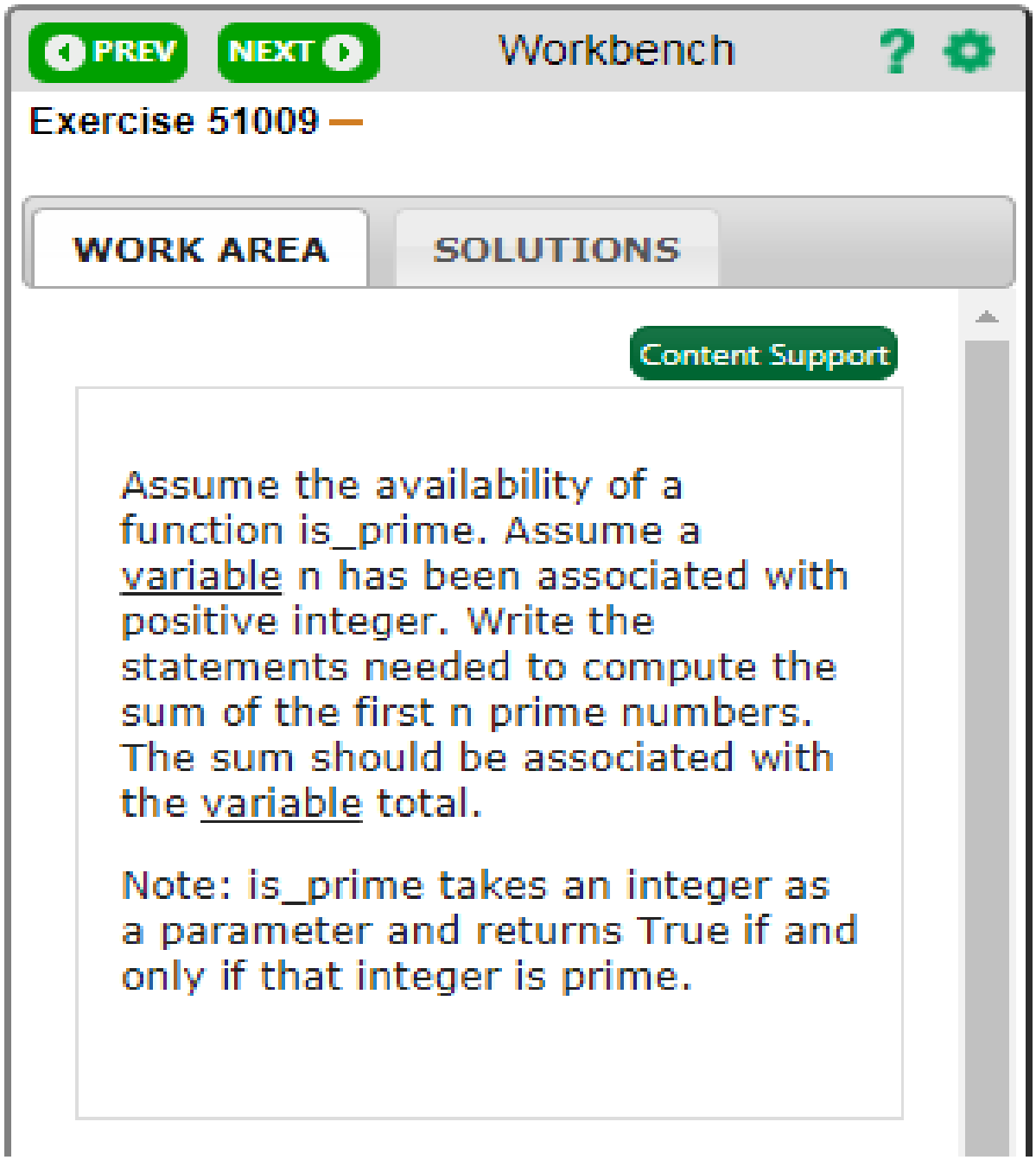
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

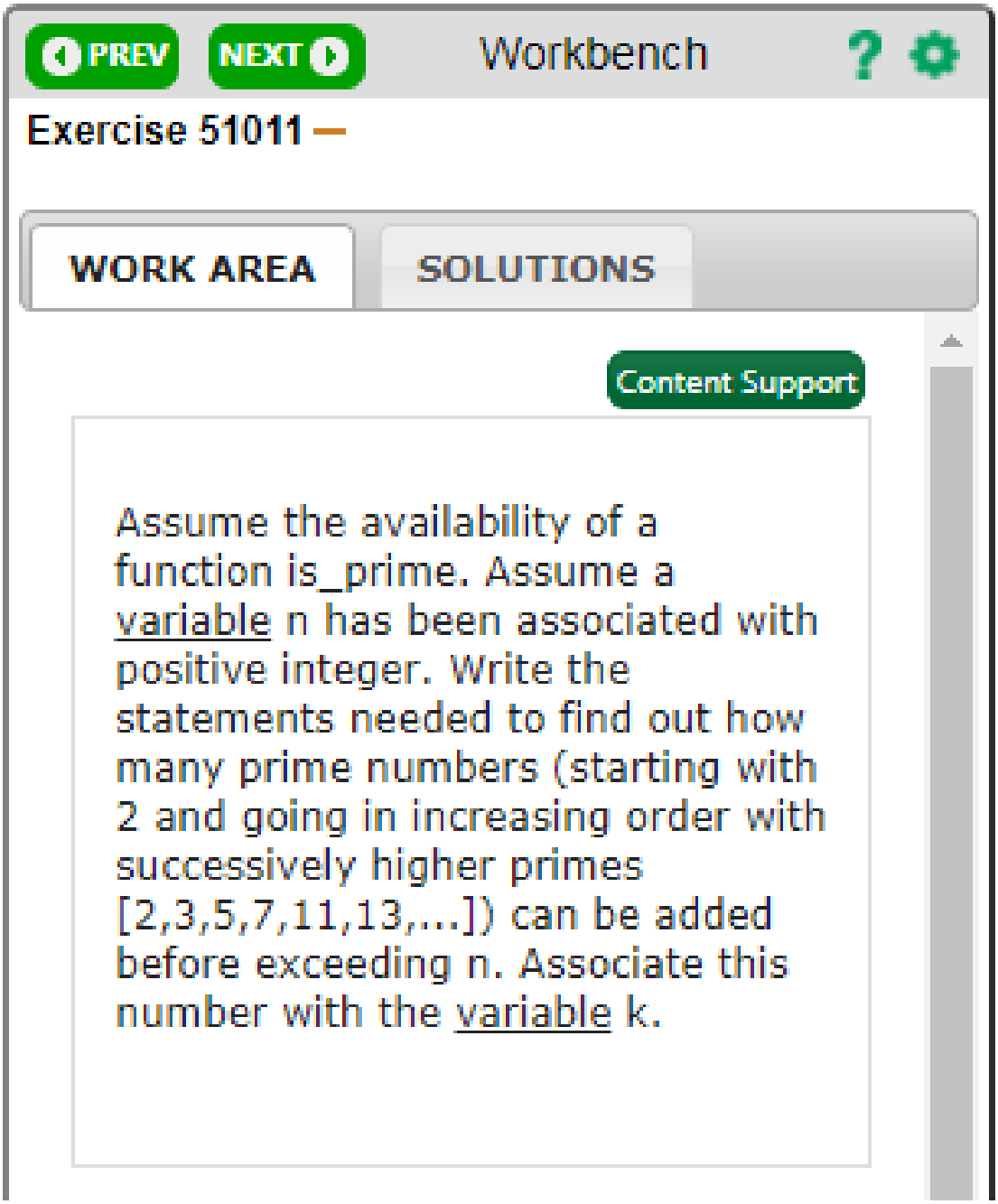
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

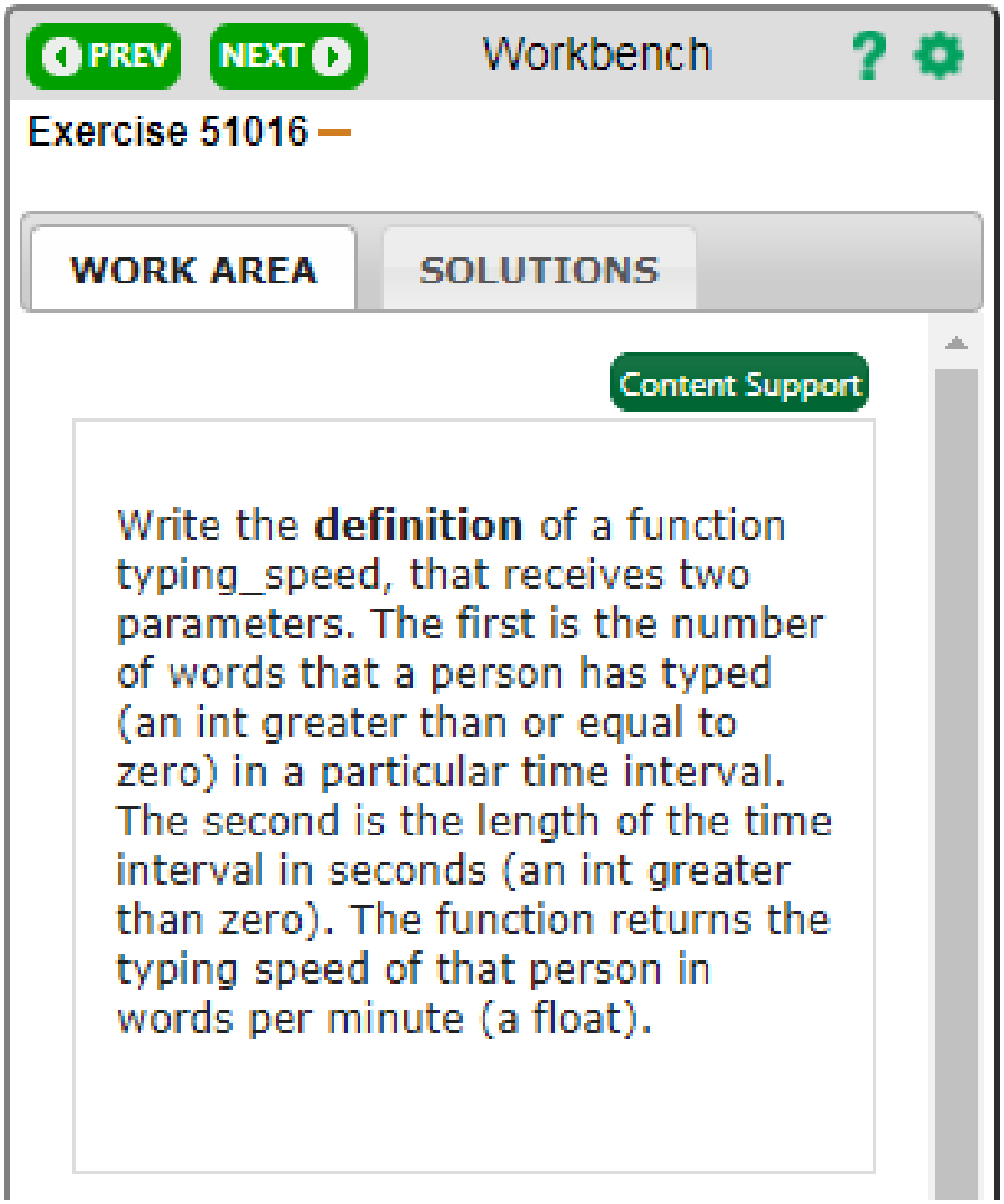
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

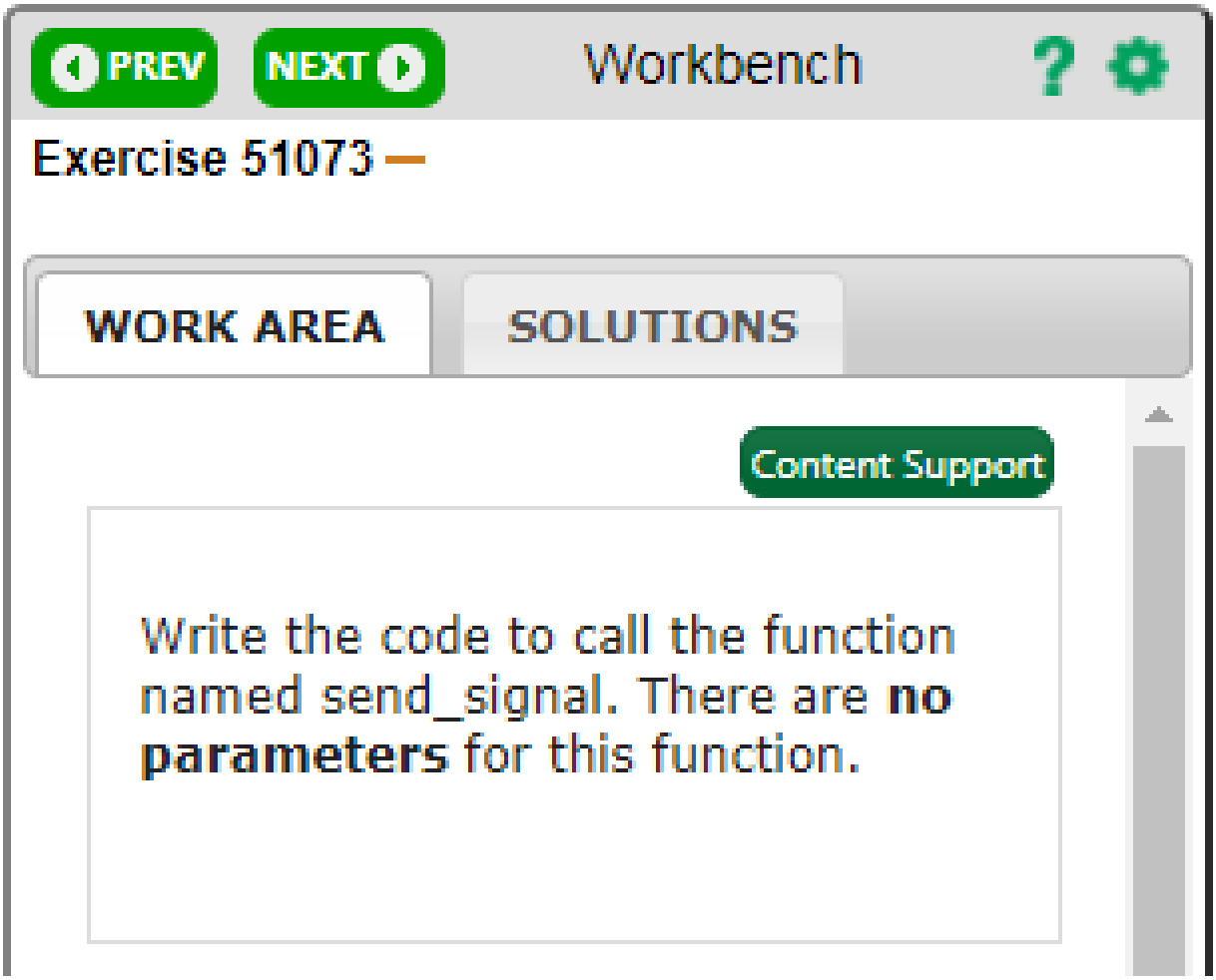
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

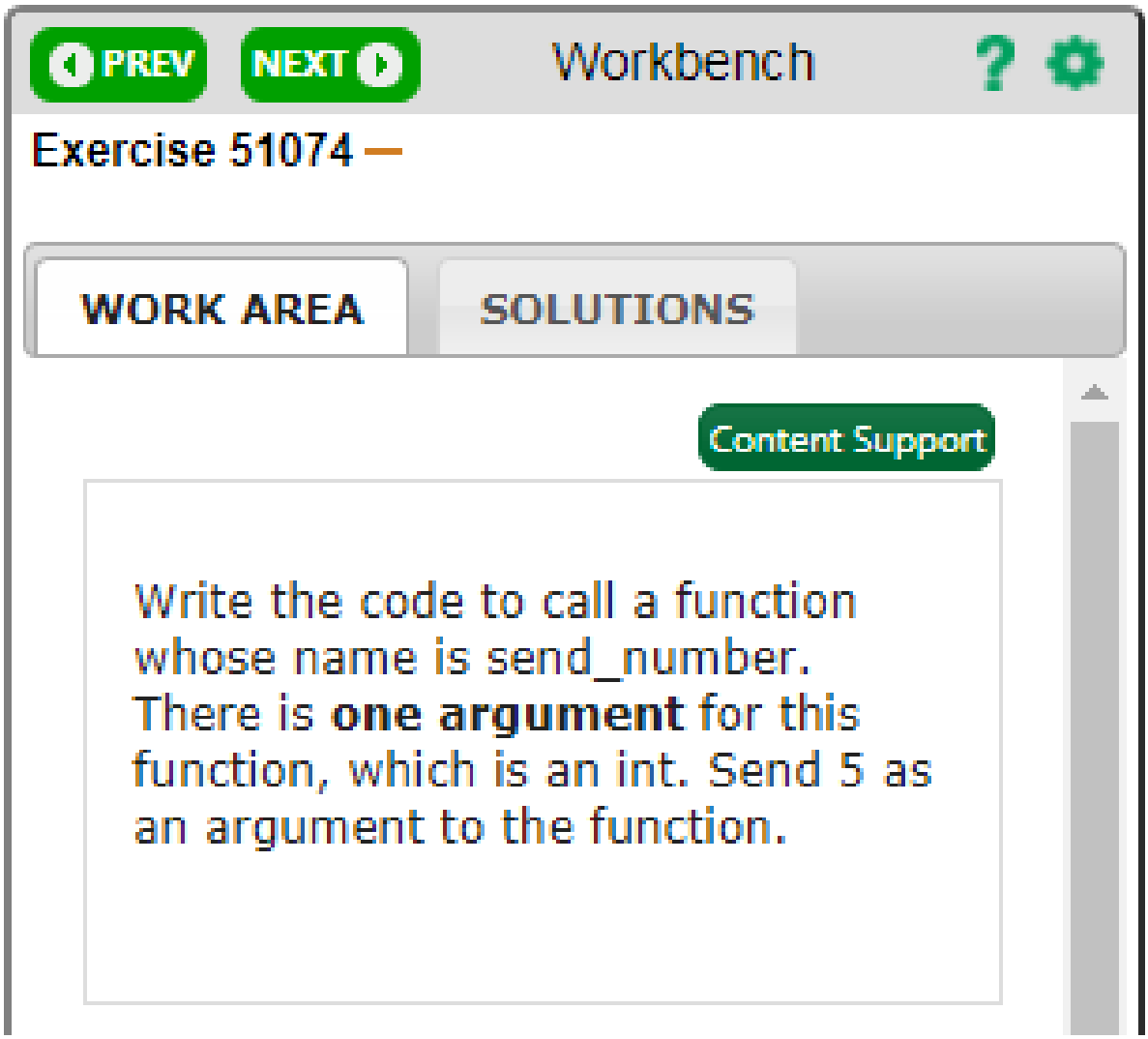
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

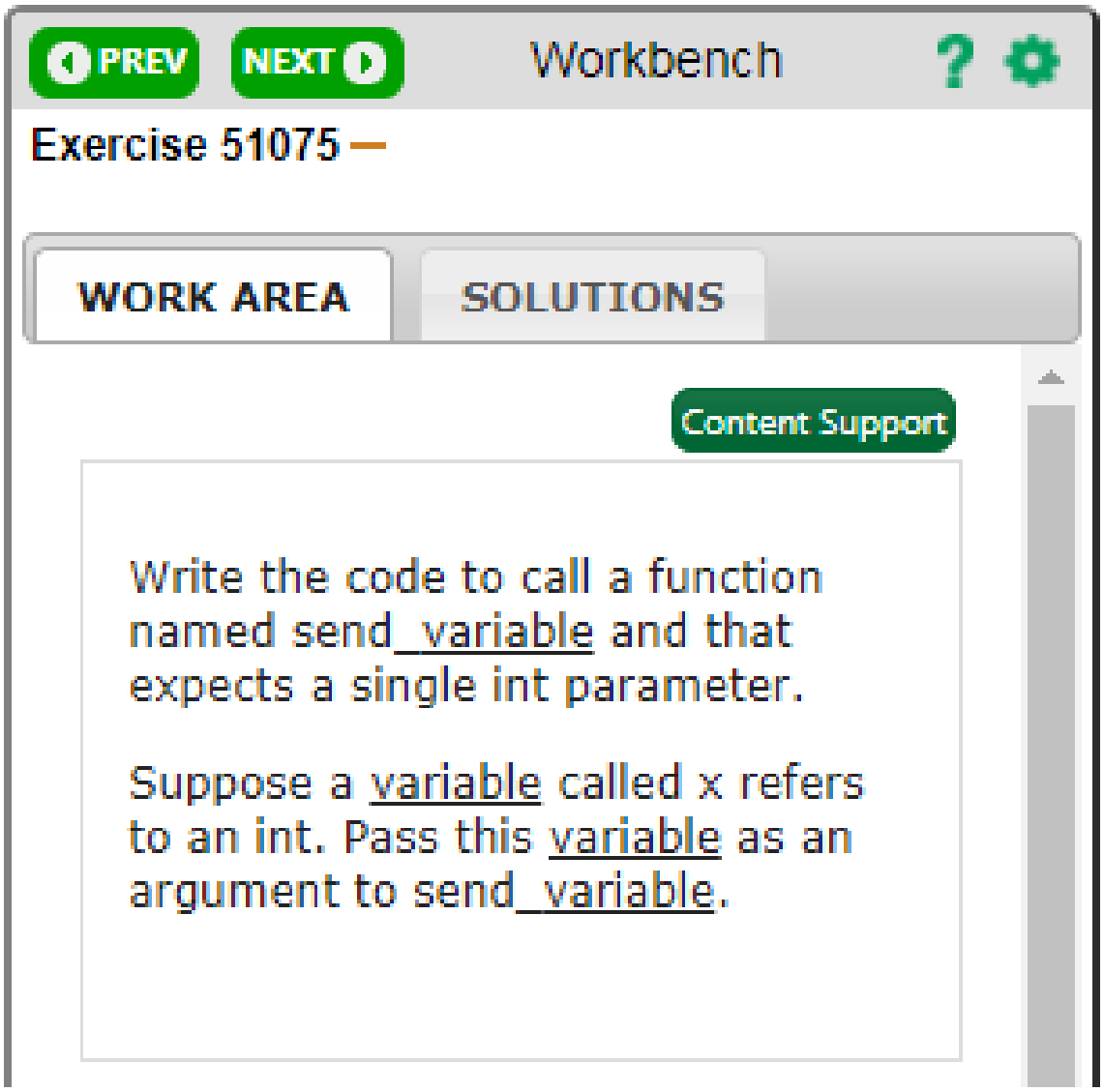
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

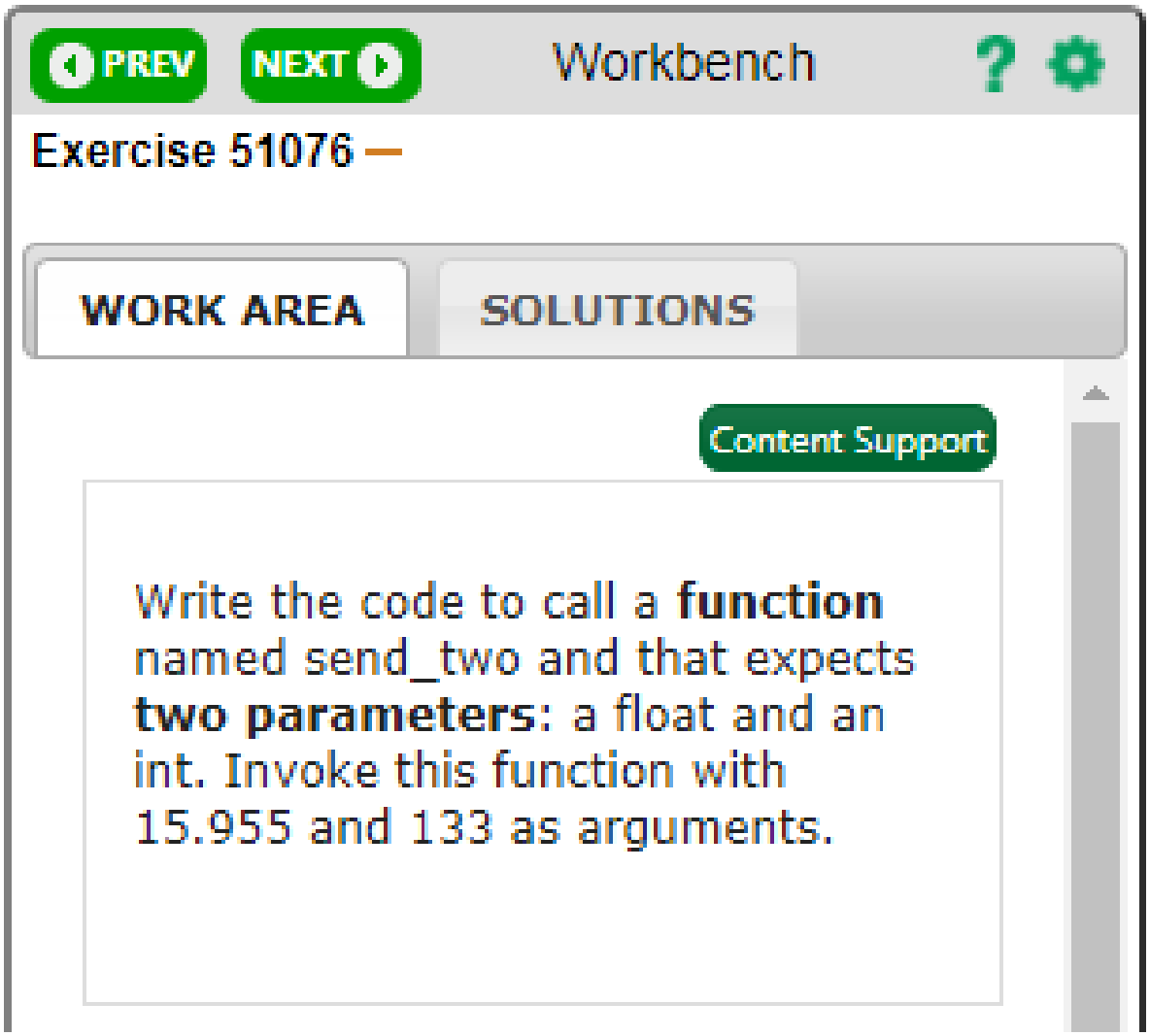
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

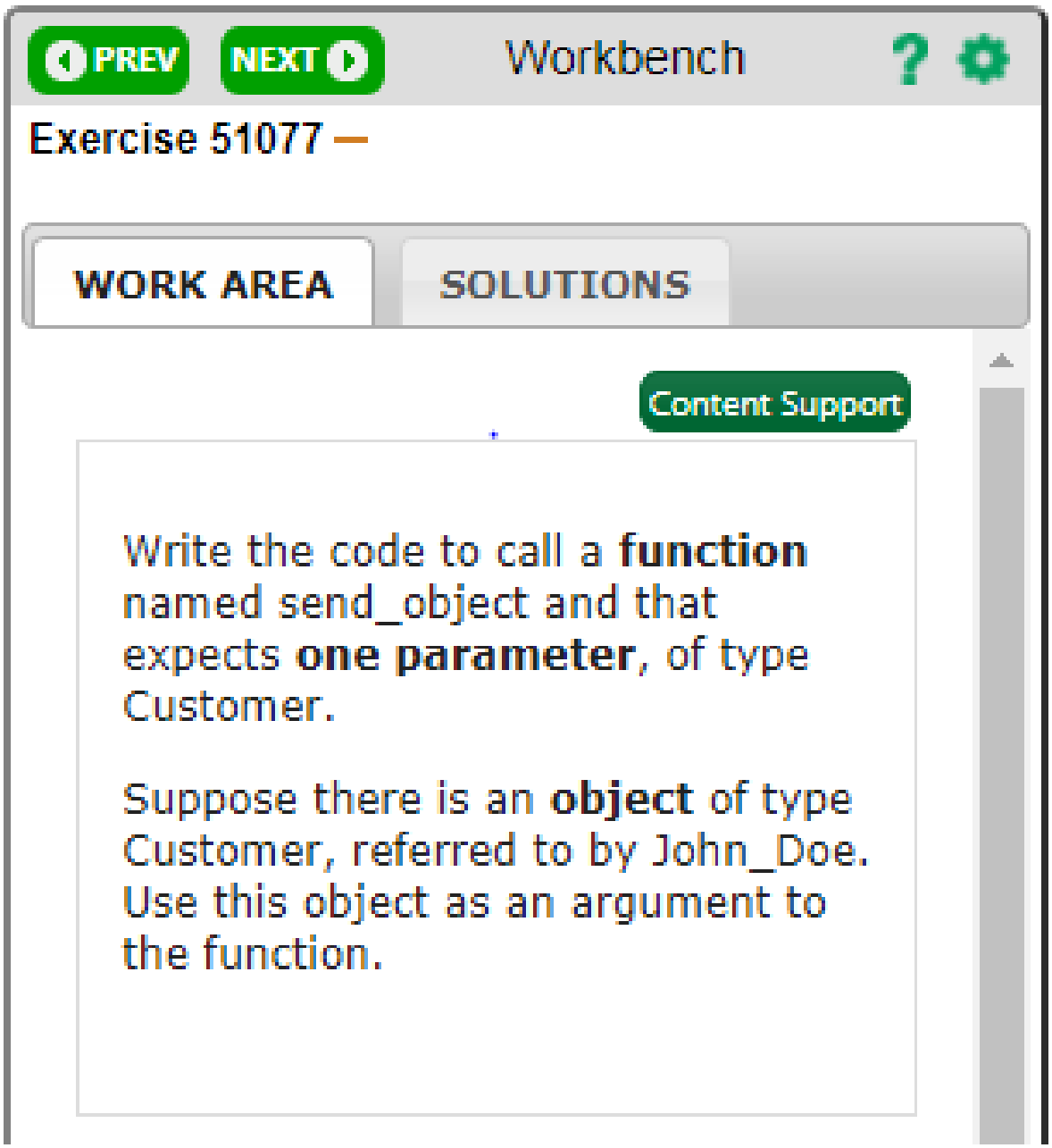
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

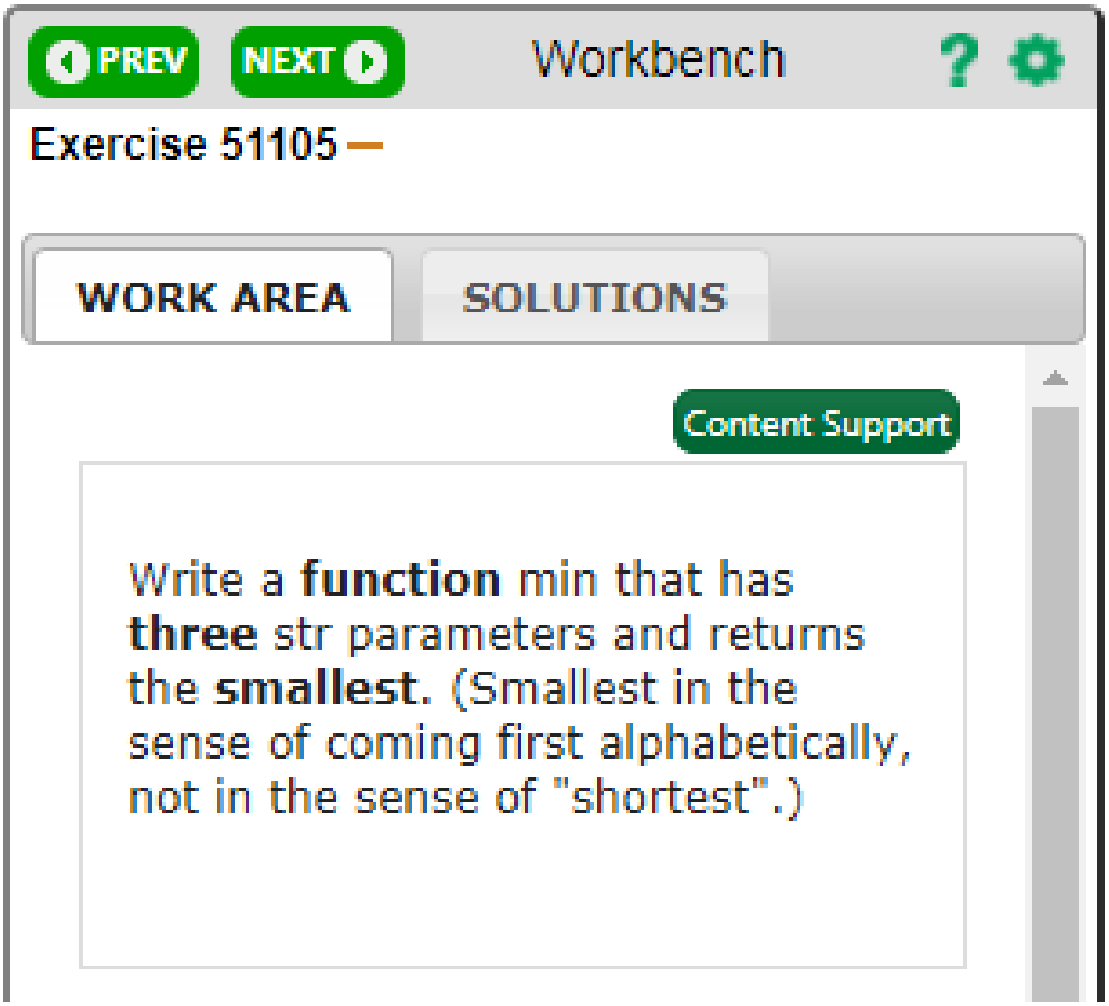
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

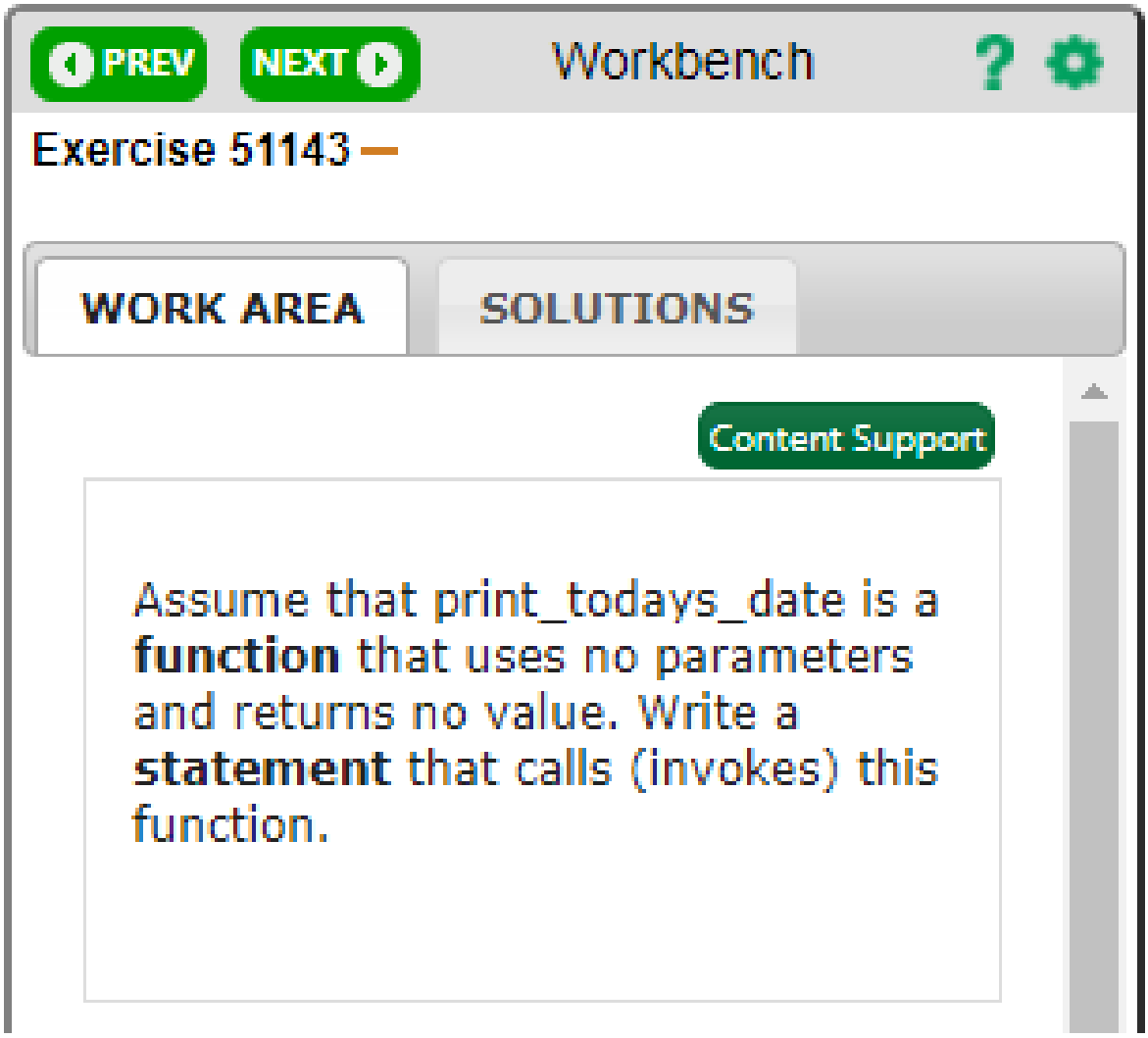
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

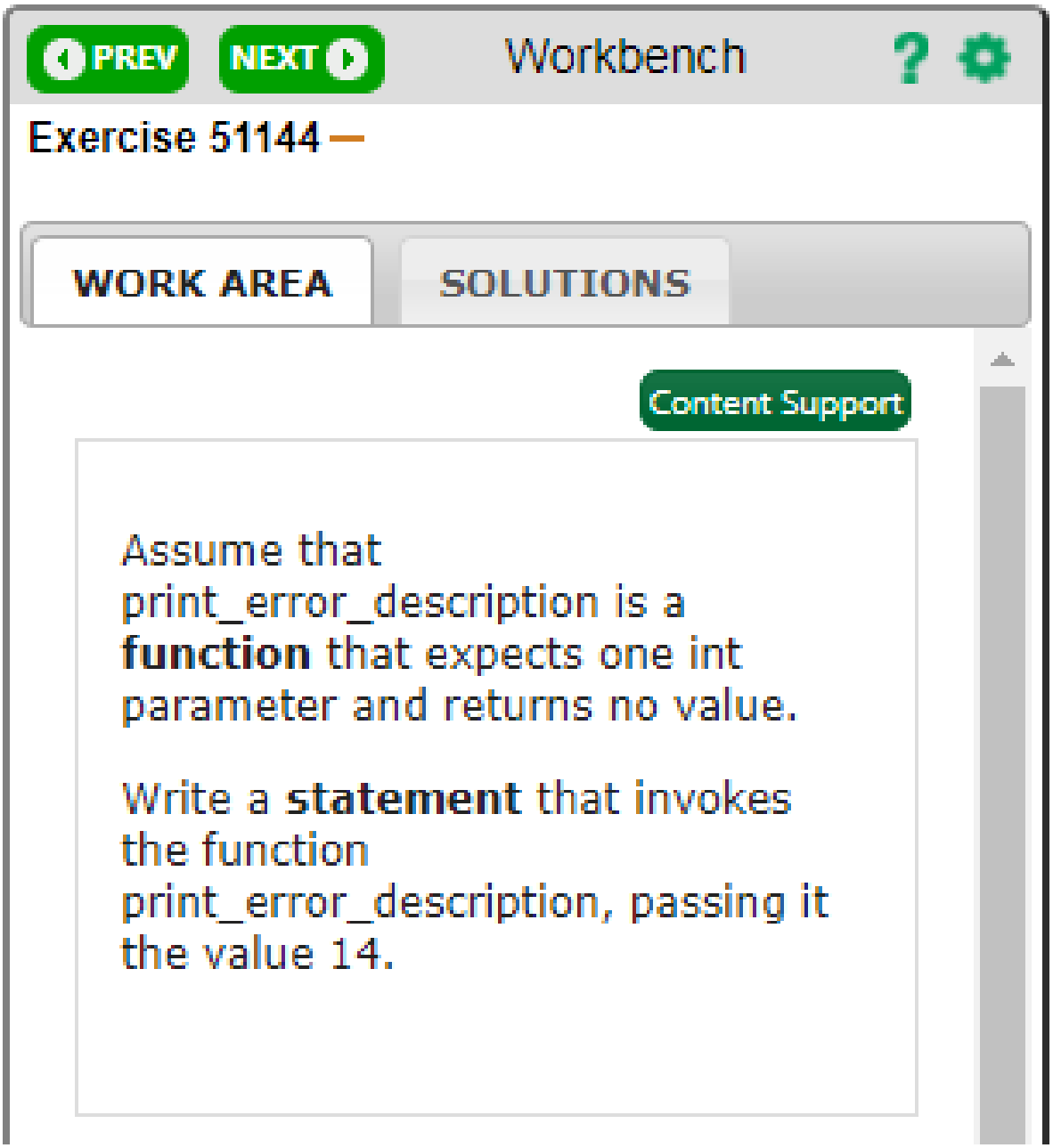
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

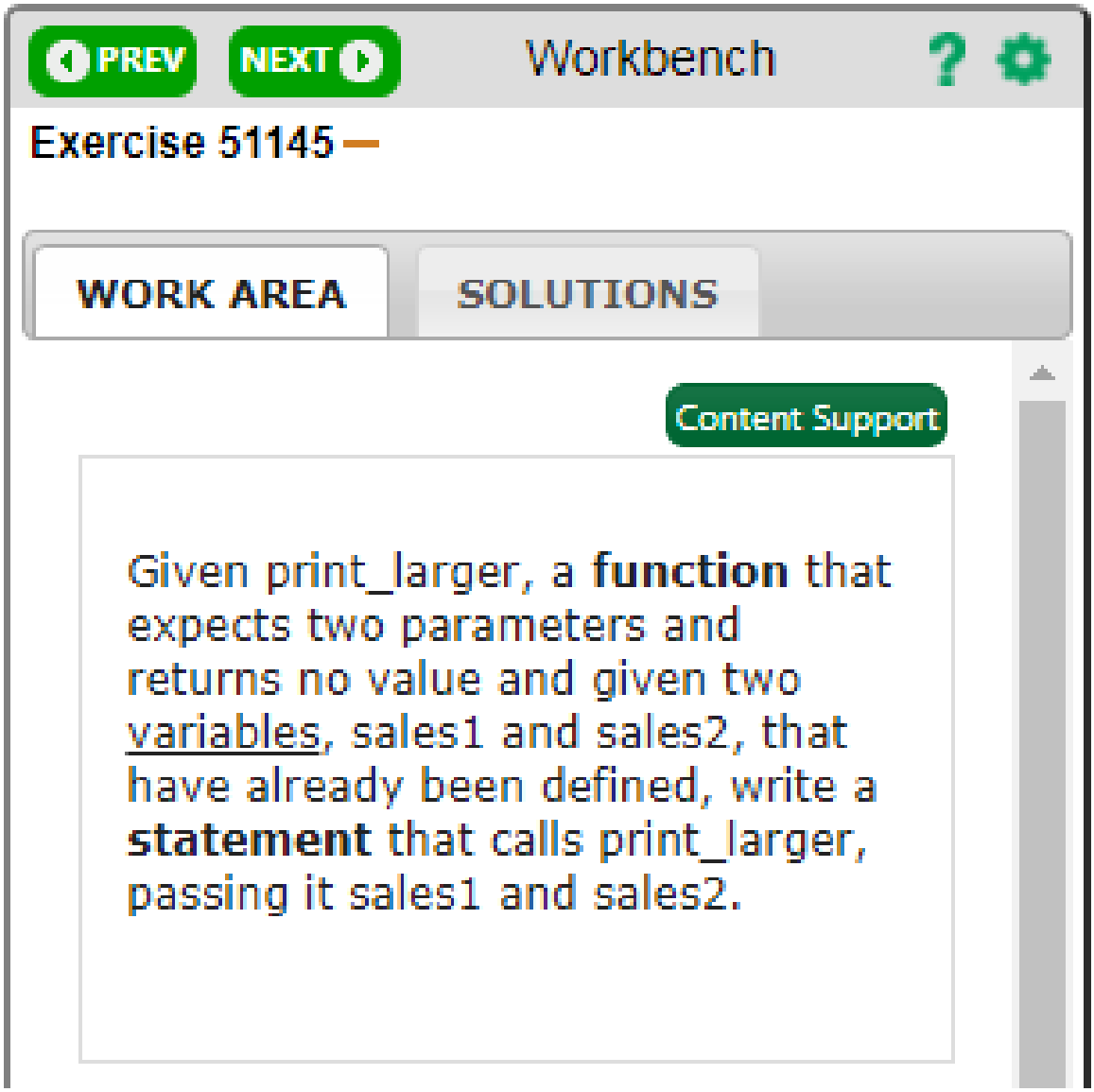
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

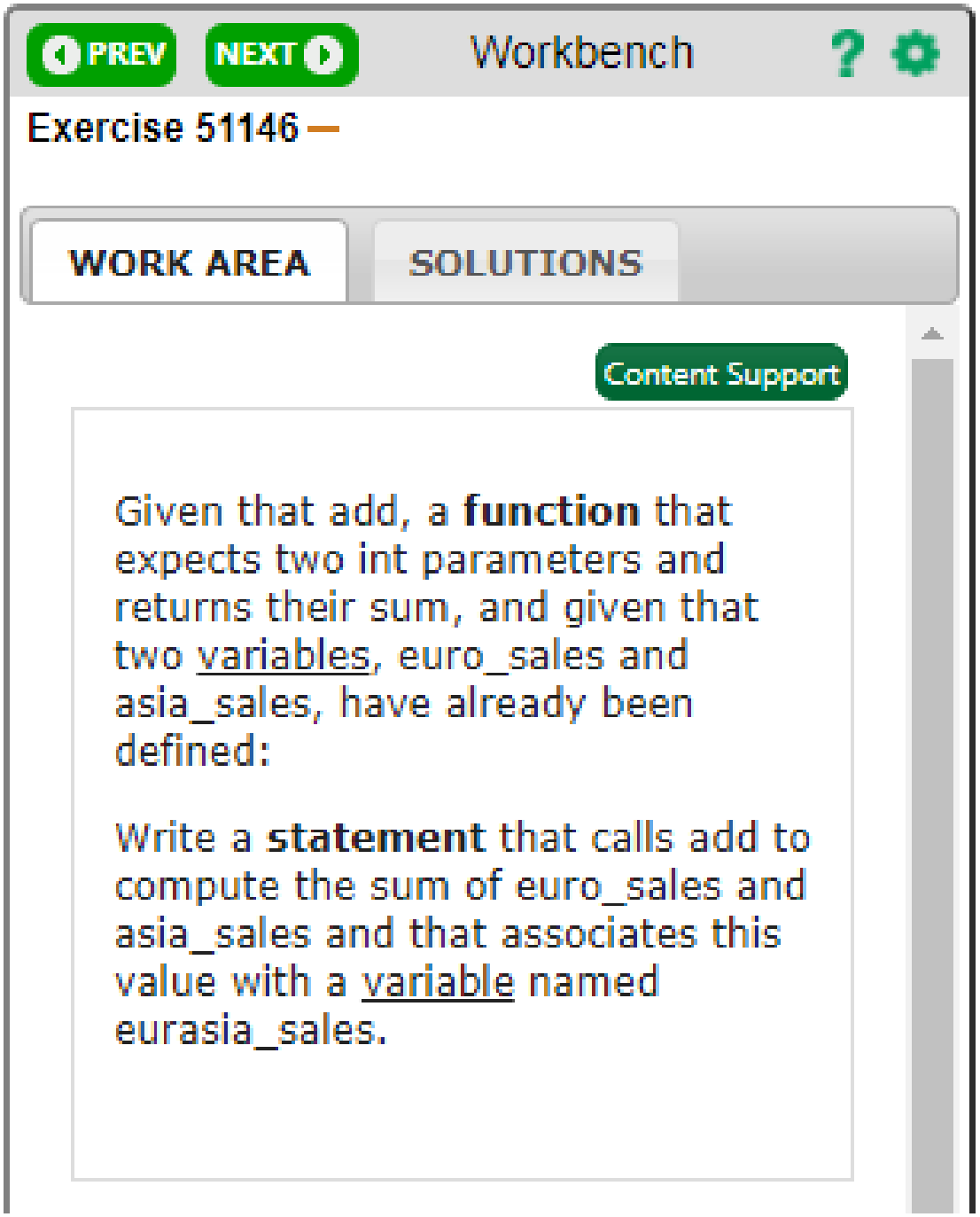
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

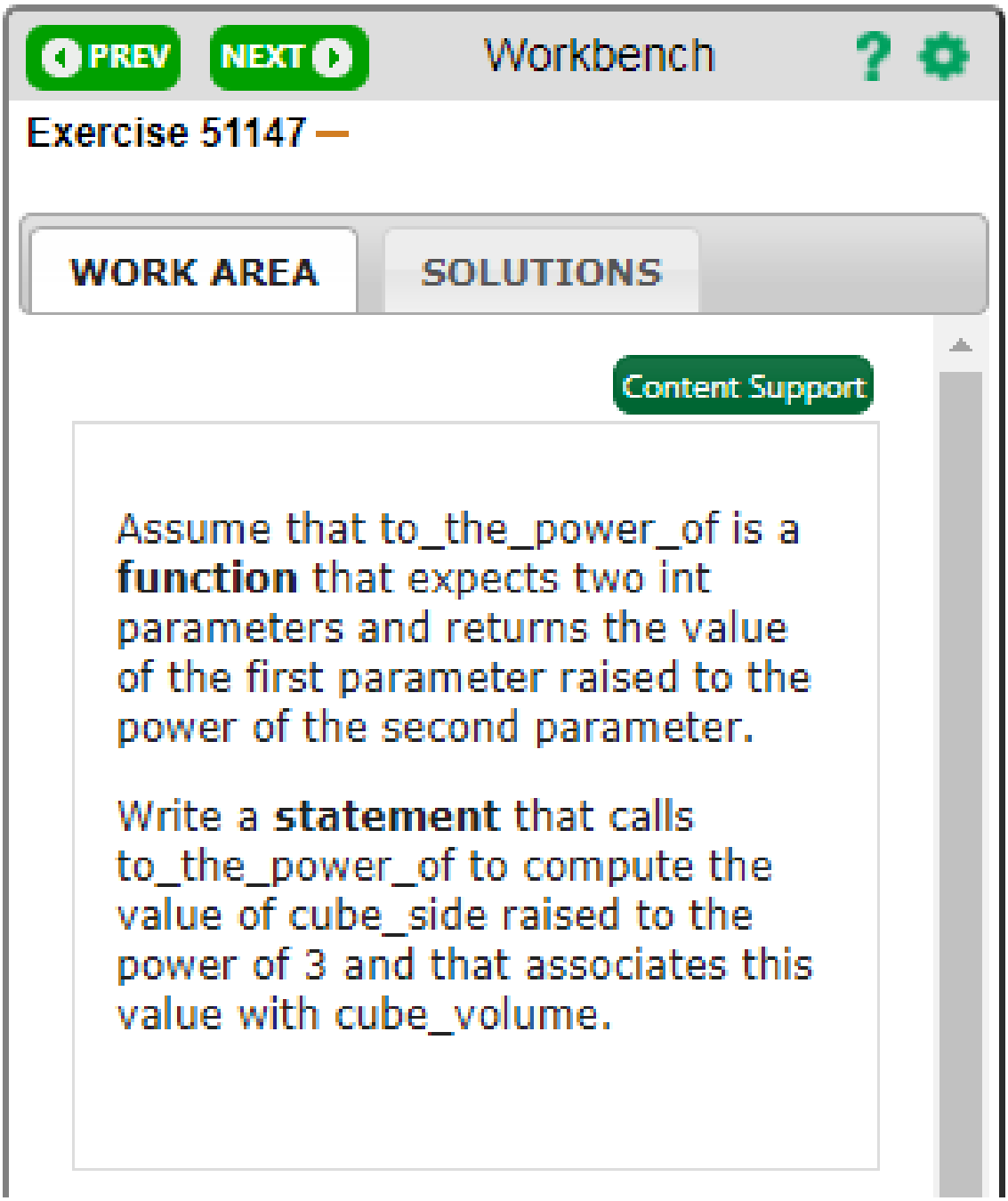
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

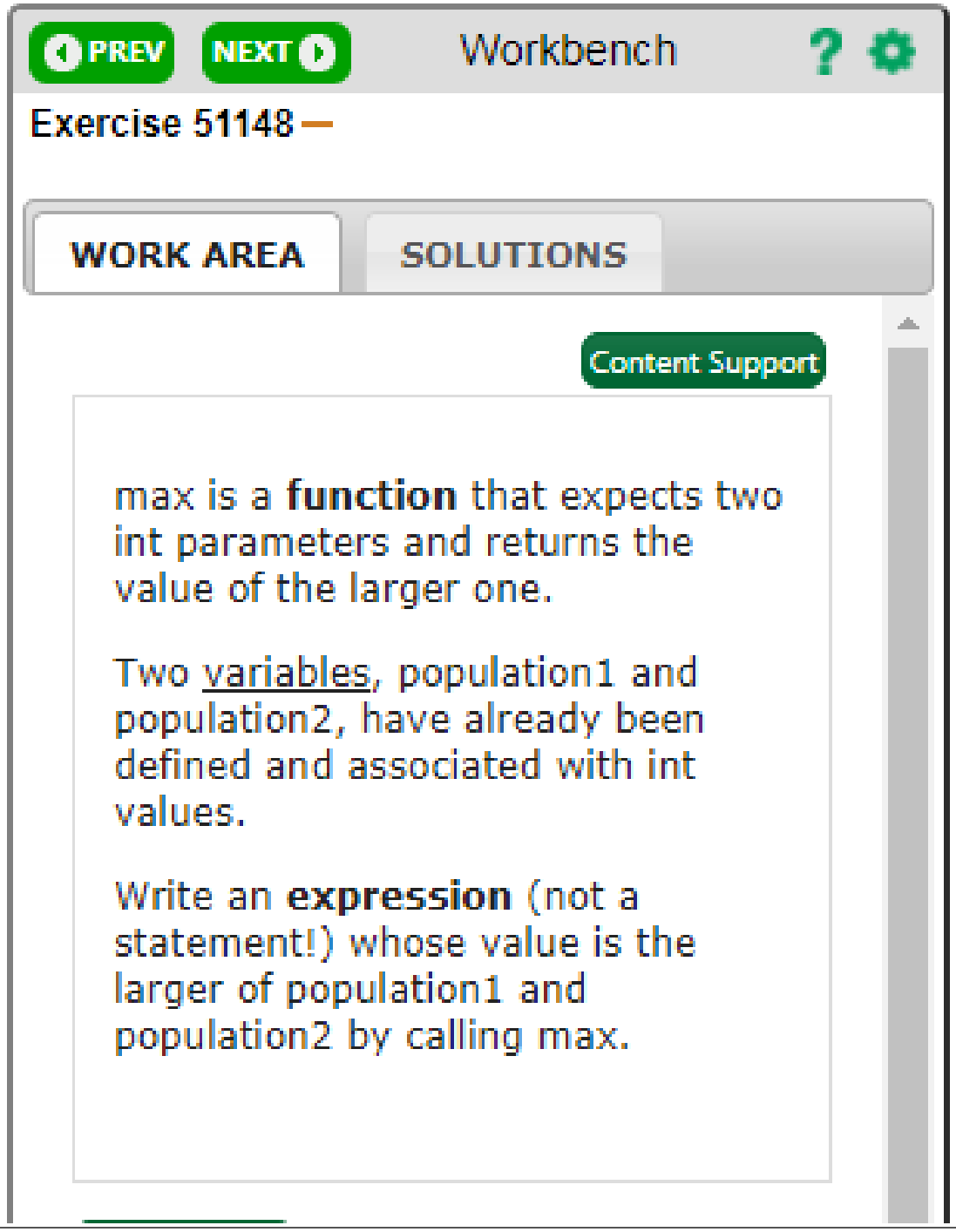
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

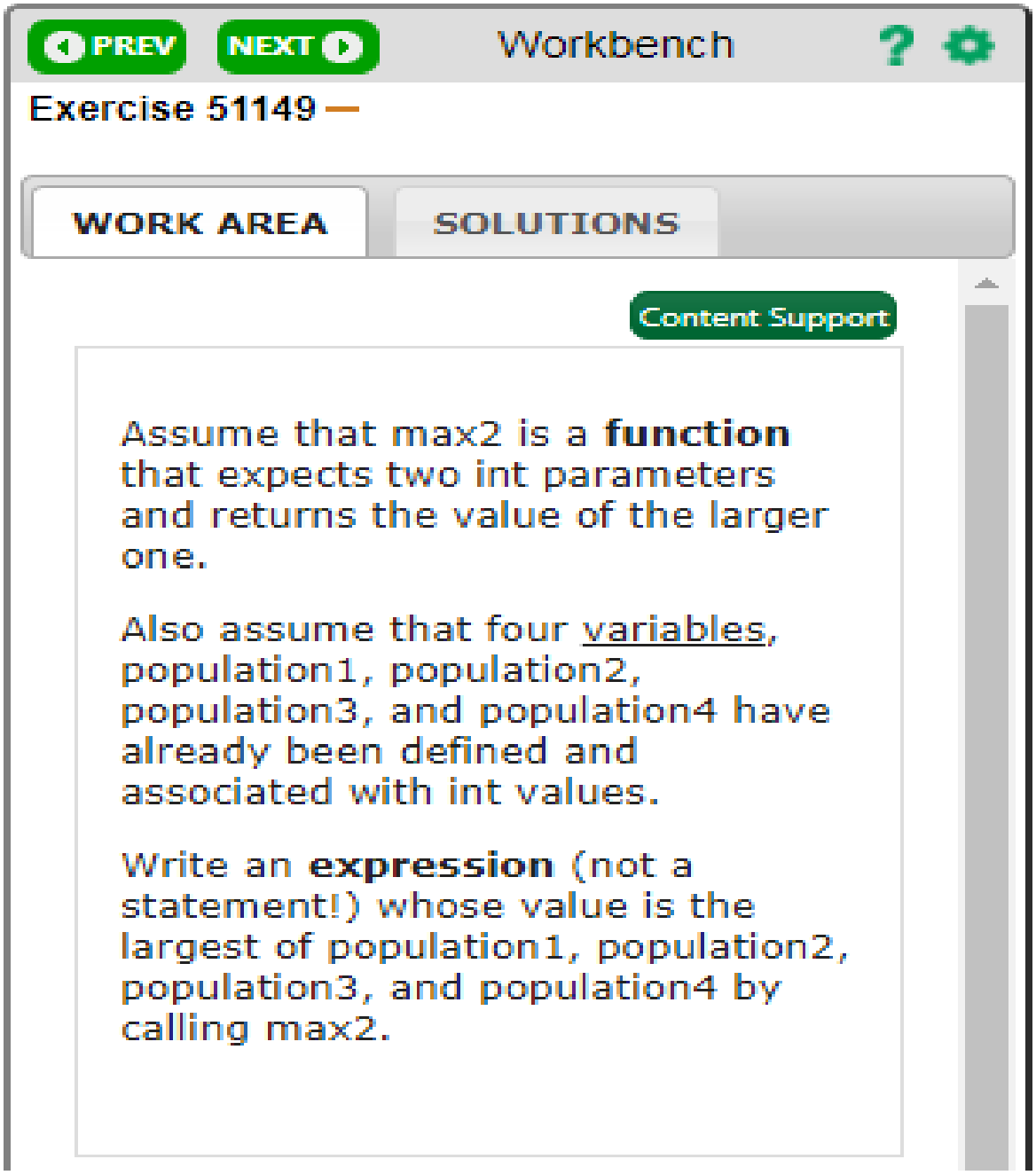
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

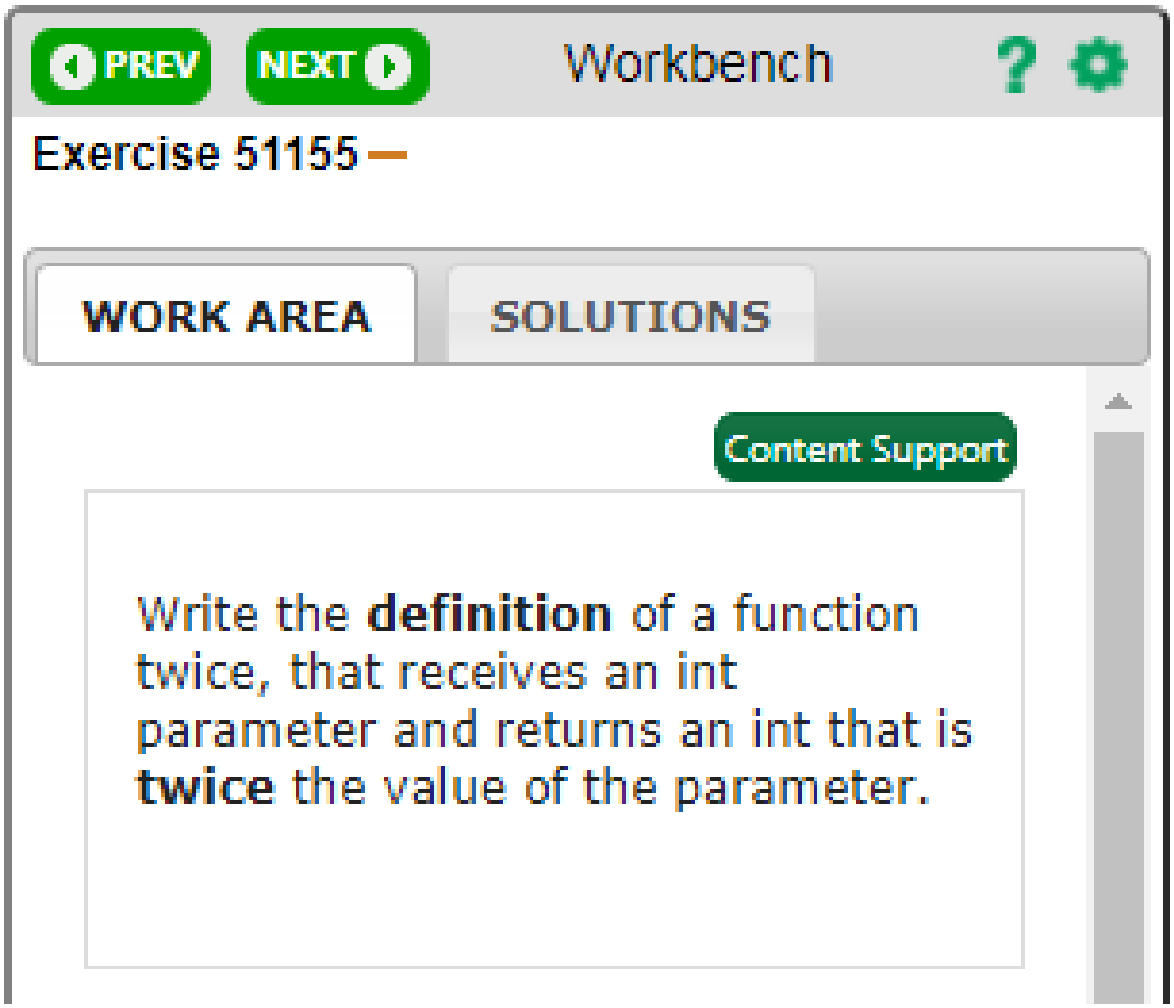
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

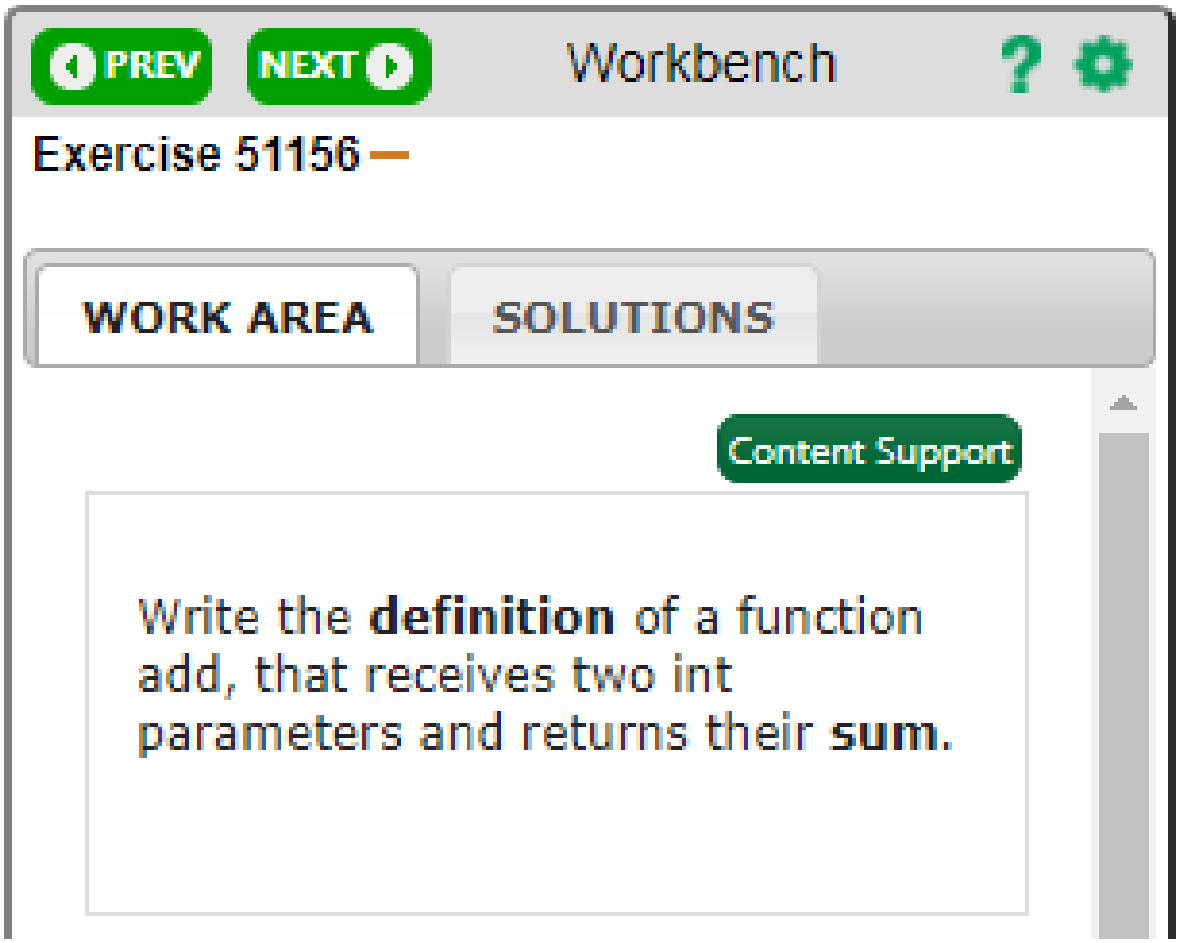
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

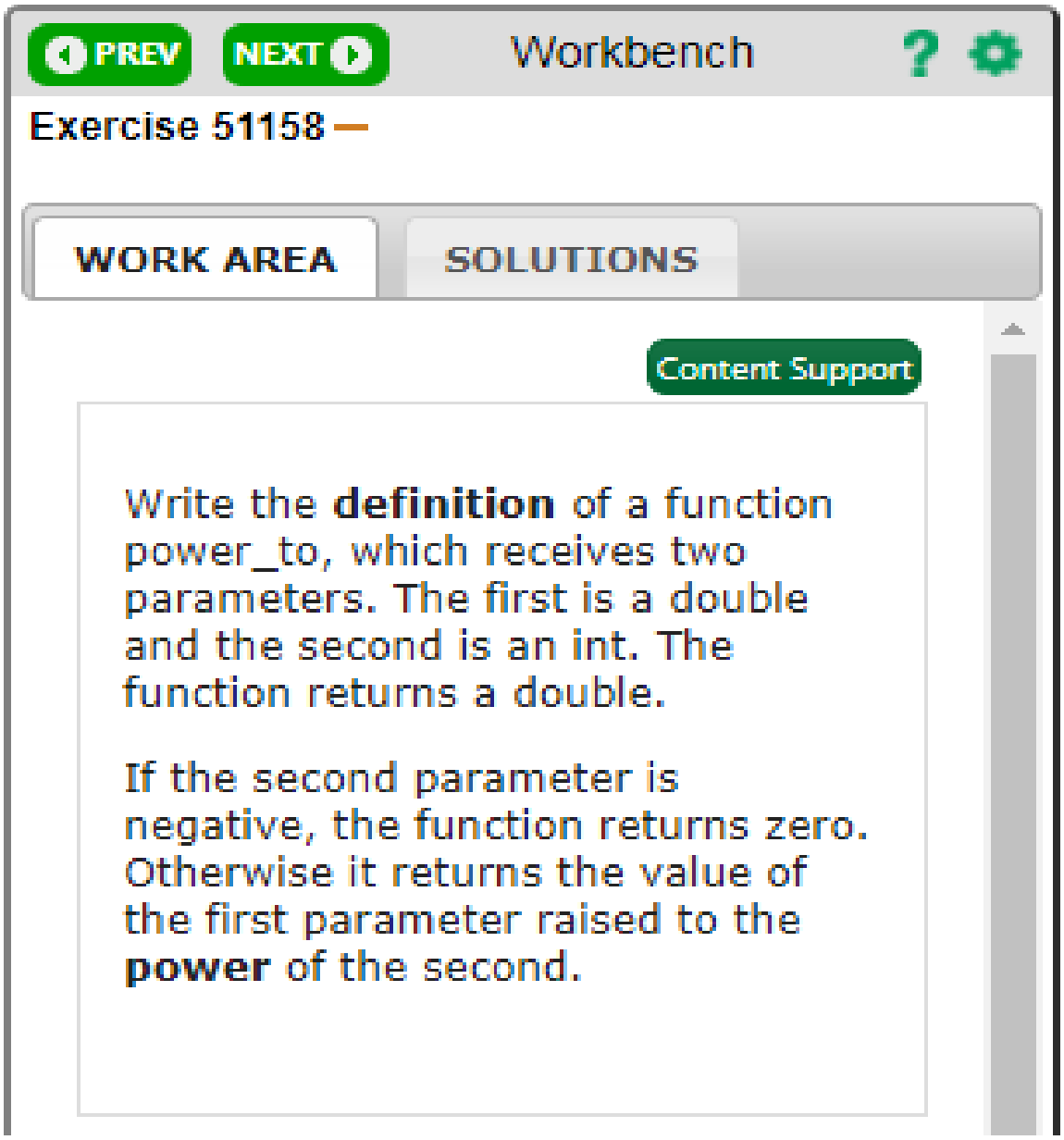
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

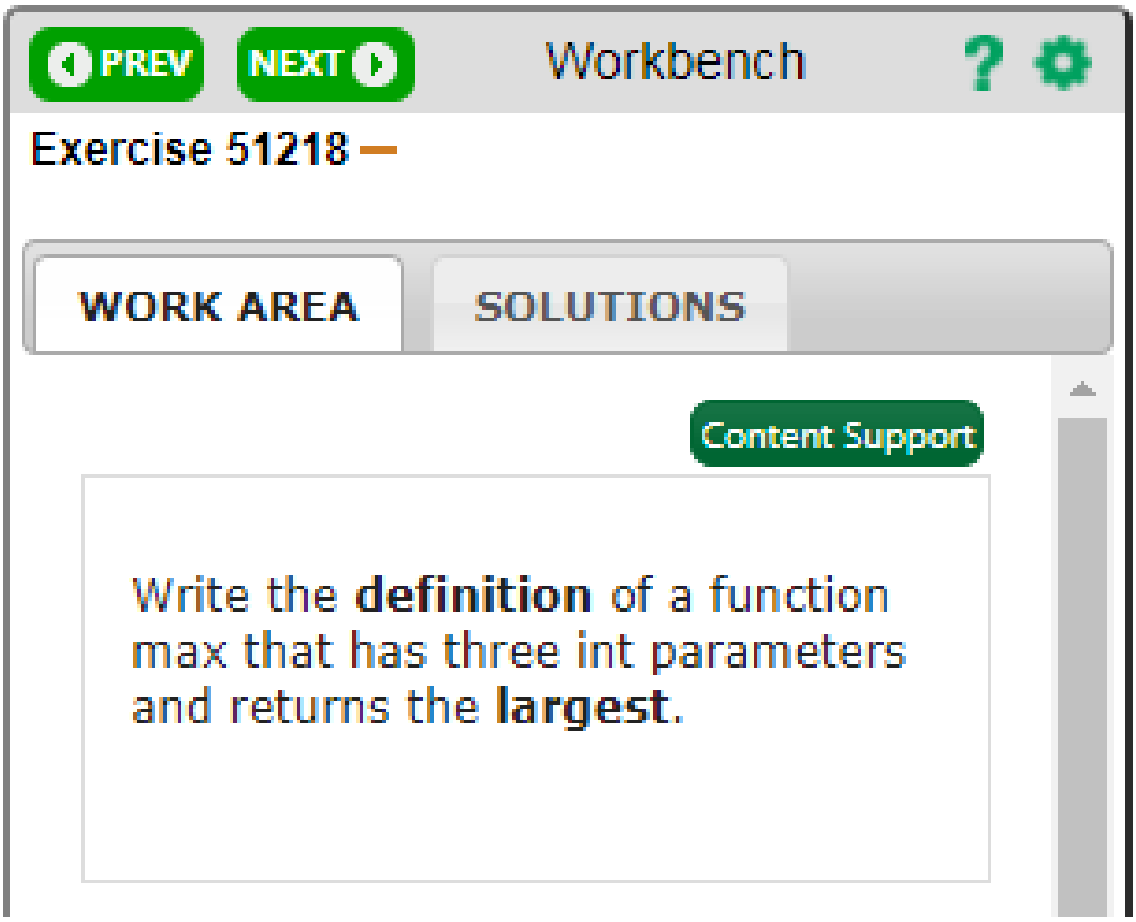
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank