

# Control Flow

# *if else Statement*

*if condition:*

*indented statement(s)*

*indented statement(s) # execution continues*

```
x = raw_input("enter number: ")
```

```
if x < 0:
```

```
    x = abs(x)
```

# *if else Statement*

*if condition:*

*indented statement(s)*

*else:*

*indented statement(s)*

```
def max_two(a, b):
    if a < b:
        return b
    else:
        return a
```

```
>>> max(10, 15)
```

```
15
```

```
>>> max('Mozzarella', 'brie')
```

```
'brie'
```

# *if else* Statement with Logical Operators

```
def test_increase(a, b, c):  
    if a < b and b < c:  
        return True  
    else:  
        return False
```

```
>>> test_increase(2, 6, 4)
```

```
False
```

```
>>> test_increase('BRIE', 'BRie', 'brie')
```

```
True
```

# Nested *if–elif–else* Statement

```
if condition_1:  
    <statements_block_A>  
  
elif condition_2:  
    <statements_block_B>  
  
elif condition_3:  
    pass  
  
else:  
    <statements_block_C>  
    <statements_block_D>
```

- when do we execute A and D?
- when do we execute B and D?
- when do we execute C?
- when do we execute D?

# Nested *if-elif-else* Statement

*if condition\_1:*

*statement\_1*

*elif condition\_2:*

*statement\_2*

*else:*

*default*

```
def compute_grade(score):
    if score > 90:
        return 'A'
    elif score > 80:
        return 'B'
    else:
        return 'C'
```

>>> **compute\_grade(85)**

**‘B’**

# Python Loop Construct

- Python has two loop constructs:

```
for <iterator> in <iterable>:  
    <statements_block>
```

```
>>> for cheese in ["Mozzarella", "Brie"]  
    —     print cheese
```

```
while <test>:  
    <statements_block_1>  
    <statements_block_2>
```

```
>>> sum = 0; counter = 0  
>>> while counter < 1000:  
    —         sum = sum + counter  
    —         counter = counter + 1
```

# *for Loop with range*

```
# ratio of consecutive Fibonacci numbers  
fib_numbers = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]  
for i in range(1, len(fib_numbers)):  
    x = fib_numbers[i]  
    y = fib_numbers[i-1]  
    print (i, float(x)/y)
```

- how do we change the flow of execution?
  - 1) continue
  - 2) break

# Python *continue* Statement

```
for <iterator> in <iterable>:  
    <statements_block_1>  
    if <test>:  
        continue  
    <statements_block_2>  
    <statements_block_3>
```

```
# ratio of consecutive odd Fibonacci numbers  
fib_numbers = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]  
for i in range(1, len(fib_numbers)):  
    x = fib_numbers[i]  
    y = fib_numbers[i-1]  
    if (x%2 ==0) or (y%2==0):  
        continue  
    print (i, float(x)/y)
```

# *break* Statement

```
for <iterator> in <iterable>:  
    <statements_block_1>  
    if <test>:  
        break  
    <statements_block_2>  
    <statements_block_3>
```

- if test is true, we exit loop and execute

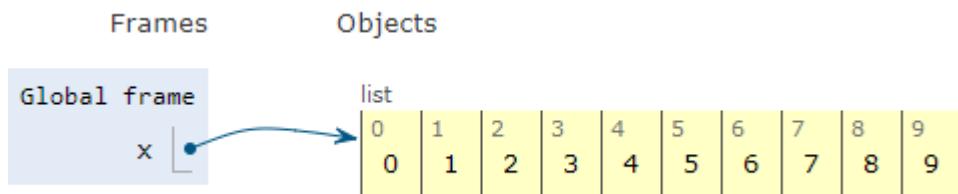
```
# print first fib. Number > 25  
  
fib_numbers = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]  
for i in range(1, len(fib_numbers)):  
  
    x = fib_numbers[i]  
  
    if (x > 25):  
  
        break  
  
    print ("first fibonacci number: ", x)
```

# *range()* Examples

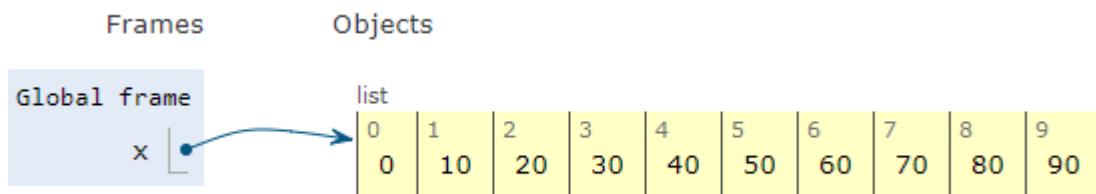
- format: *range(start, stop[, step])*
- default step is 1, default start 0

```
>>> x = range(0,10,1)
```

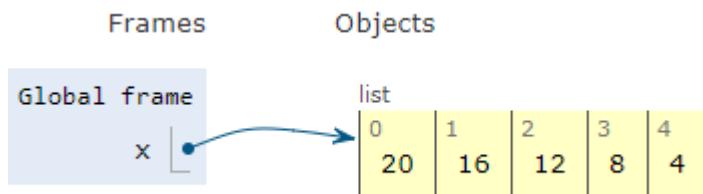
```
>>> x = range(10)           # shorthand
```



```
>>> x = range(0, 100, 10)  # count by 10
```



```
>>> x = range(20, 0, -4)  # backwards by 4
```



# *range()* Example

```
>>> x = range(10)
```

```
>>> sum = 0
```

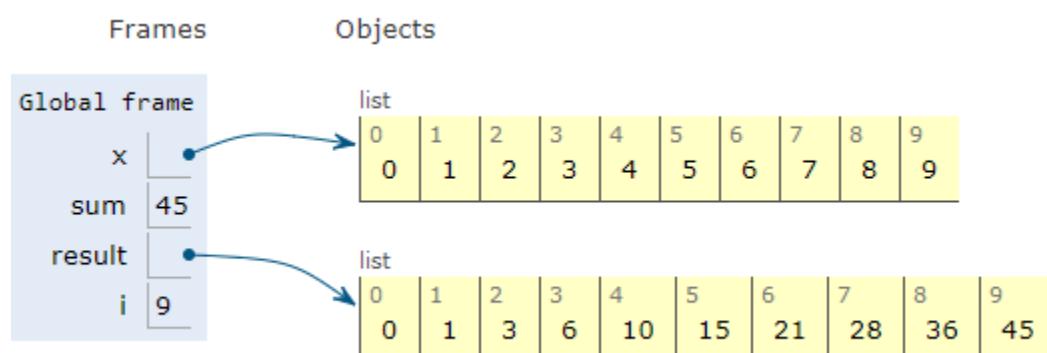
```
>>> list = []
```



```
>>> for i in x:
```

```
...                          sum = sum + i
```

```
...                          result.append(sum)
```



```
>>> print(result)
```

```
[0, 1, 3, 6, 10, 15, 21, 28, 36, 45]
```

# Counting with range() Construct

- Python has two loop constructs:

```
for <iterator> in <iterable>:  
    <statements_block>
```

```
>>> for cheese in ["Mozzarella", "Brie"]  
    —     print cheese
```

```
while <test>:  
    <statements_block_1>  
    <statements_block_2>
```

```
>>> sum = 0; counter = 0  
>>> while counter < 1000:  
    —         sum = sum + counter  
    —         counter = counter + 1
```

# Review Problems

# Interview Problem

- what is `pass` statement used for?

# Interview Problem

- what are the two major loop statements?

# Interview Problem

- explain the use of break and continue in looping

# Interview Problem

- when would you use a *break* statement in a *for* loop?

# Interview Problem

- what is the structure of a *for* loop?

# Interview Problem

- what is the structure of a *while* loop?

# Interview Problem

- under what circumstances would you use a `while` statement rather than `for`?

# Interview Problem

- when would you use a *continue* statement in a *for* loop?

# Interview Problem

- which statement of Python is used whenever a statement is required syntactically but the program needs no action?

# Interview Problem

- what are the optional statements to use inside a <try – except> block?

# Interview Problem

- print sum of numbers from 1 to 100

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' in green rounded rectangles, followed by a 'Workbench' button in blue and red, and a help/gear icon. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A 'Content Support' button is located above a large text area. The main text area contains the following question:

Given the variables name1 and name2, write a fragment of code that assigns the **larger** of their associated values to first.

(NOTE: "larger" here means alphabetically larger, not "longer". Thus, "mouse" is larger than "elephant" because "mouse" comes later in the dictionary than "elephant"!)

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, and 'Workbench' in the center. To the right of 'Workbench' are two icons: a question mark and a gear. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. The 'WORK AREA' tab is currently selected. In the main content area, there is a text box containing the following instruction:

Write a **conditional** that assigns True to `fever` if `temperature` is greater than 98.6.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' in green rounded rectangles, followed by a 'Workbench' button in blue and orange, and a help icon with a question mark and a gear symbol. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A 'Content Support' button is located above a large text area. The main text area contains the following instruction:

Write a **conditional** that assigns  
10,000 to bonus if the goods\_sold  
is **greater** than 500,000.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' in green rounded rectangles, followed by a 'Workbench' button in blue and orange, and a help icon ('?') and settings icon ('⚙️'). Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A 'Content Support' button is located above a large text area. The main text area contains the following instruction:

Write a **conditional** that **decreases** the value associated with `shelf_life` by 4 if the value associated with `outside_temperature` is **greater** than 90.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a help icon with a gear and question mark on the right. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A 'Content Support' button is located above a scrollable area containing the following text:

Write a **conditional** that **multiplies** the value associated with **pay** by **one-and-a-half** if **worked\_overtime** is associated with **True**.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a question mark icon and gear icon on the right. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A green button labeled 'Content Support' is located above the main text area. The main text area contains the following challenge description:

Write an `if/else` statement that compares `age` with 65, adds 1 to `senior_citizens` if `age` is greater than or equal to 65, and adds 1 to `non_seniors` otherwise.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a question mark icon and a gear icon on the right. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A green button labeled 'Content Support' is located above the main text area. The main text area contains the following instruction:

Write an **if/else** statement that compares `sold_yesterday` and `sold_today`, and based upon that comparison assigns `sales_trend` the value -1 (the case where `sold_yesterday` is greater than `sold_today`) or 1.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a question mark icon and a gear icon on the right. Below these is the title 'Exercise 51120'. Underneath the title are two tabs: 'WORK AREA' and 'SOLUTIONS', with 'WORK AREA' being the active tab. A 'Content Support' button is located below the tabs. The main area contains the following text:

Write an **if/else** statement that  
assigns True to fever if temperature  
is greater than 98.6; otherwise it  
assigns False to fever.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there is a toolbar with 'PREV' and 'NEXT' buttons, a 'Workbench' tab, and a help/gear icon. Below the toolbar, the title 'Exercise 51121 —' is displayed. There are two tabs: 'WORK AREA' (selected) and 'SOLUTIONS'. A 'Content Support' button is located above the main content area. The main content area contains the following text:

Write an **if/else** statement that adds 1 to minors if age is **less than** 18, adds 1 to adults if age is 18 through 64 and adds 1 to seniors if age is 65 or older.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a question mark icon and a gear icon on the right. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A 'Content Support' button is located above a large text area. The main text area contains the following instructions:

Given that `ph` refers to a float, write a **statement** that compares `ph` to 7.0 and makes the following assignments (RESPECTIVELY!) to the variables `neutral`, `base`, and `acid`:

- 0,0,1 if `ph` is **less** than 7
- 0,1,0 if `ph` is **greater** than 7
- 1,0,0 if `ph` is **equal** to 7

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' (with a left arrow icon) and 'NEXT' (with a right arrow icon). To the right of these are links for 'Workbench', '?', and a gear icon. Below the navigation is the title 'Exercise 51160 —'. Underneath the title are two tabs: 'WORK AREA' and 'SOLUTIONS'. A green button labeled 'Content Support' is located above the main content area. The main content area contains the following text: 'Given the variables x, y, and z, each associated with an int, write a fragment of code that assigns the smallest of these to min.'

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a help/gear icon on the right. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A 'Content Support' button is located above the main text area. The main text area contains the following challenge description:

Use two variables k and total to write a for loop to compute the sum of the squares of the first 50 counting numbers, and store this value in total. Thus, your code should put  $1*1 + 2*2 + 3*3 + \dots + 49*49 + 50*50$  into total. Use no variables other than k and total.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a help icon with a gear symbol on the right. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A green button labeled 'Content Support' is located above the main text area. The main text area contains the following instruction:

Given a variable n refers to a positive int value, use two additional variables, k and total to write a for loop to compute the sum of the **cubes** of the first n counting numbers, and store this value in total. Thus your code should put  $1*1*1 + 2*2*2 + 3*3*3 + \dots + n*n*n$  into total. Use no variables other than n, k, and total.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a help icon ('?') and settings icon ('⚙️') on the right. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A green button labeled 'Content Support' is located above the main content area. The main content area contains the following text:

Use the variables k and total to write a while loop that computes the sum of the squares of the first 50 counting numbers, and associates that value with total. Thus your code should associate  $1*1 + 2*2 + 3*3 + \dots + 49*49 + 50*50$  with total. Use no variables other than k and total.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a help icon with a gear symbol on the right. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A green button labeled 'Content Support' is located near the bottom of the tabs. The main area contains the following text:

Given that `n` refers to a positive int  
use a while loop to compute the  
**sum of the cubes** of the first `n`  
counting numbers, and associate  
this value with `total`. Use no  
variables other than `n`, `k`, and `total`.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a programming exercise interface. At the top, there are navigation buttons: 'PREV' and 'NEXT' in green rounded rectangles, followed by 'Workbench' in a grey box, and a help icon ('?') and settings icon ('⚙️') in green. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A vertical scroll bar is on the right side of the main content area. The main content area contains the following text:

In this exercise, use the following variables: `i`, `lo`, `hi`, and `result`. Assume that `lo` and `hi` each are associated with an int and that `result` refers to 0.

Write a `while` loop that adds the integers from `lo` up through `hi` (inclusive), and associates the sum with `result`.

Your code should not change the values associated with `lo` and `hi`. Also, just use these variables: `i`, `lo`, `hi`, and `result`.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' in green rounded rectangles, followed by a 'Workbench' button in blue and red, and a help/gear icon. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A 'Content Support' button is located above a large text area. The main text area contains the following question:

Given `x` and `y`, each associated with an int, write a fragment of code that associates the **larger** of these with another variable named `max`.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a software interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a question mark icon and gear icon on the right. Below these is the title 'Exercise 51250 —'. Underneath the title are two tabs: 'WORK AREA' and 'SOLUTIONS', with 'WORK AREA' being the active tab. To the right of the tabs is a green button labeled 'Content Support'. The main content area contains the following text:

Assume there is a variable, h already associated with a positive integer value. Write the code necessary to compute the sum of the first h perfect squares, starting with 1. (A perfect square is an integer like 9, 16, 25, 36 that is equal to the square of another integer (in this case 3\*3, 4\*4, 5\*5, 6\*6 respectively).) Associate the sum you compute with the variable q. For example, if h is 4, you would assign 30 to q because the first 4 perfect squares (starting with 1) are: 1, 4, 9, 16 and  $30 = 1 + 4 + 9 + 16$ .

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a question mark icon and a gear icon on the right. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A green button labeled 'Content Support' is located above the main text area. The main text area contains the following problem statement:

Assume there is a variable, h already associated with a positive integer value. Write the code necessary to compute the sum of the perfect squares whose value is less than h, starting with 1. (A perfect square is an integer like 9, 16, 25, 36 that is equal to the square of another integer (in this case 3\*3, 4\*4, 5\*5, 6\*6 respectively).) Associate the sum you compute with the variable q. For example, if h is 19, you would assign 30 to q because the perfect squares (starting with 1) that are less than h are: 1, 4, 9, 16 and  $30 = 1 + 4 + 9 + 16$ .

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a software interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT'. To the right of these are links for 'Workbench', '?', and a gear icon. Below the navigation is the title 'Exercise 51252 —'. Underneath the title are two tabs: 'WORK AREA' and 'SOLUTIONS'. A green button labeled 'Content Support' is located above the main content area. The main content area contains the following text:

Assume there is a variable, `h` already associated with a positive integer value. Write the code necessary to count the number of perfect squares whose value is less than `h`, starting with 1. (A perfect square is an integer like 9, 16, 25, 36 that is equal to the square of another integer (in this case 3\*3, 4\*4, 5\*5, 6\*6 respectively).) Assign the sum you compute to a variable `q`. For example, if `h` is 19, you would assign 4 to `q` because there are perfect squares (starting with 1) that are less than `h` are: 1, 4, 9, 16.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a software interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a help icon ('?') and settings icon ('⚙️') on the right. Below the navigation is the title 'Exercise 51253 —'. Underneath the title are two tabs: 'WORK AREA' (selected) and 'SOLUTIONS'. A green button labeled 'Content Support' is located below the tabs. The main area contains the following text:

Assume there are two variables, k and m, each already associated with a positive integer value and further assume that k's value is smaller than m's. Write the code necessary to compute the number of perfect squares between k and m. (A perfect square is an integer like 9, 16, 25, 36 that is equal to the square of another integer (in this case 3\*3, 4\*4, 5\*5, 6\*6 respectively).) Associate the number you compute with the variable q. For example, if k and m had the values 10 and 40 respectively, you would assign 3 to q because between 10 and 40 there are these perfect squares: 16, 25, and 36.,.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, and 'Workbench' in the center. To the right of 'Workbench' are two icons: a question mark and a gear. Below these buttons, the title 'Exercise 51259 —' is displayed. Underneath the title are two tabs: 'WORK AREA' and 'SOLUTIONS'. The 'WORK AREA' tab is currently selected, indicated by a grey background. In the 'WORK AREA' section, there is a large text area containing the following instruction:

Associate the average of the numbers from 1 to n (where n is a positive integer value) with the variable avg.

Below this text area is a green button labeled 'Content Support'.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a question mark icon and a gear icon on the right. Below the navigation bar, the title 'Exercise 51268' is displayed. Underneath the title, there are two tabs: 'WORK AREA' and 'SOLUTIONS'. The 'WORK AREA' tab is currently selected. In the main content area, there is a button labeled 'Content Support'. The main text area contains the following instruction:

Associate True with the variable `isAscending` if the list `numbers` is in ascending order (that is, if each element of the list is greater than or equal to the previous element in the list). Otherwise, associate False with `isAscending`

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a help icon with a gear on the right. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A green button labeled 'Content Support' is located above the main content area. The main content area contains the following text:

Given a positive integer `n`, assign  
True to `is_prime` if `n` has no factors  
other than 1 and itself.  
(Remember, `m` is a factor of `n` if `m`  
divides `n` evenly.)

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a software interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' in green rounded rectangles, followed by a 'Workbench' button in a grey rectangle, and a help icon ('?') and settings icon ('⚙️') in green rounded rectangles. Below these is the title 'Exercise 51286 —'. Underneath the title are two tabs: 'WORK AREA' and 'SOLUTIONS', with 'WORK AREA' being the active tab. A 'Content Support' button is located above the main content area. The main content area contains the following text:

An *arithmetic progression* is a sequence of numbers in which the distance (or difference) between any two successive numbers is the same. This in the sequence 1, 3, 5, 7, ..., the distance is 2 while in the sequence 6, 12, 18, 24, ..., the distance is 6.

Given the positive integer `distance` and the positive integer `n`, associate the variable `sum` with the sum of the elements of the arithmetic progression from 1 to `n` with distance `distance`. For example, if `distance` is 2 and `n` is 10, then `sum` would be associated with 25 because  $1+3+5+7+9 = 25$ .

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a software interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a question mark icon and a gear icon on the right. Below the navigation bar, the title 'Exercise 51897 —' is displayed. Underneath the title, there are two tabs: 'WORK AREA' and 'SOLUTIONS'. The 'WORK AREA' tab is currently selected. In the main content area, there is a text box containing the following problem statement:

Assume that `isIsosceles` is a boolean variable, and that the variables `isoCount`, `triangleCount`, and `polygonCount` have all been initialized. Write a statement that adds 1 to each of these count variables (`isoCount`, `triangleCount`, and `polygonCount`) if `isIsosceles` is true.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**