Name: **Date:** **Class:**

Creating Functions in Python With Copilot Worksheet Answer Key

Introduction

Instructions
Before beginning this worksheet, ensure that you have downloaded all necessary materials and have read the Getting Started with Visual Studio Code and Copilot document.

Introduction
Human programmers often struggle with complexity. To manage this, we break down large problems into smaller, more solvable parts using functions. Functions are essential in software design, performing single tasks and making code easier to read, test, and debug.

In this worksheet, you will learn how to create functions in Python using Copilot and understand what makes a reasonable task for Copilot to handle.

Components of a Function
Every function in Python has:

1. **Function Header (Signature)**: This includes the def keyword, the function name, and its inputs.
2. **Function Body**: The code that defines what the function does.
3. **Return Statement**: The value that the function gives back.

Example
Here's a basic function that finds the smaller of two numbers:

```
# This function takes two numbers and returns the smaller one
def smaller(a, b):

      if a < b:
         return a
      else:
    return b
```

Name: **Date:** **Class:**

Exercise 1: Understanding Function Headers

Instructions: Read the following function headers and answer the following questions:

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

1. What is the name of the function?
   Name: add

2. What are the inputs to the function?
   Inputs: a, b

3. What does the function return?
   Returns: a + b

```
def greet(name):
    return "Hello, " + name
```

4. What is the name of the function?
   Name: greet

5. What is the input to the function?
   Input: name

6. What does the function return?
   Returns: "Hello, " + name

Name: **Date:** **Class:**

Exercise 2: Writing Function Headers

Instructions: Write the function headers for the following tasks:

1. A function named *multiply* that takes two inputs, *x* and *y*.
   def multiply(x, y):

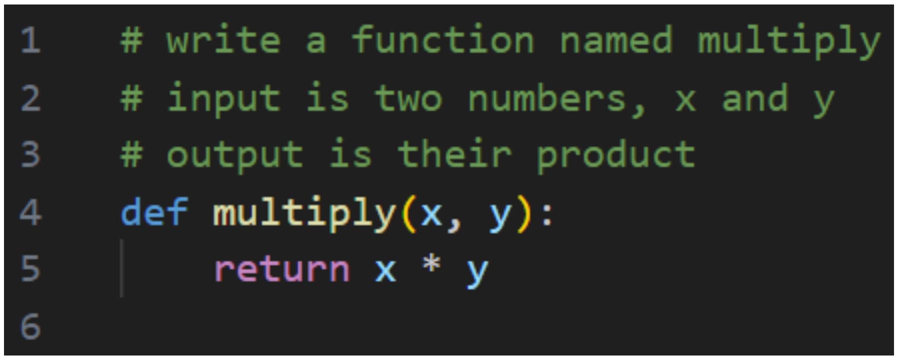
2. A function named *is_even* that takes one input, *num*.
   def is_even(num):

3. A function named *max_of_three* that takes three inputs, *a*, *b*, and *c*.
   def max_of_three(a, b, c):

Using Copilot To Write Functions
Copilot can help you write functions. Use comments (#'s) to guide Copilot on what you want the function to do. Follow the example below.

Example function: Write a function named *multiply* that takes two inputs, *x* and *y*, and returns their product.

Example answer using Copilot:



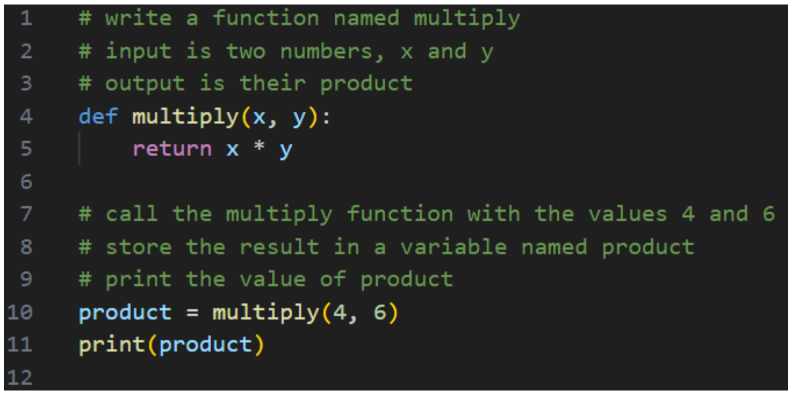
```
1    # write a function named multiply
2    # input is two numbers, x and y
3    # output is their product
4    def multiply(x, y):
5        return x * y
6
```

Lines 1-3 are comments you write. Lines 4 and 5 are the code generated by Copilot.

Name: **Date:** **Class:**

Using Copilot To Call Functions
Once we have a function, how do we use it? To use a function, we need to call it. Calling a function means invoking it with specific parameter values, known as arguments. Each value in Python has a type, and we must ensure we provide values of the correct type. For instance, our function expects two numbers; if we provide non-numeric values, it may not work as expected. When we call a function, it executes its code and returns a result. To use this result later, we need to capture it in a variable, which is simply a name that refers to a value.

Calling the multiply function using Copilot:



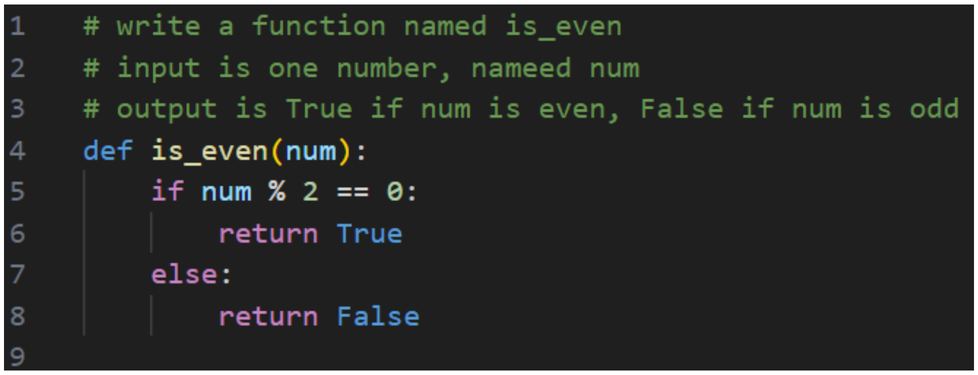
```
1   # write a function named multiply
2   # input is two numbers, x and y
3   # output is their product
4   def multiply(x, y):
5       return x * y
6
7   # call the multiply function with the values 4 and 6
8   # store the result in a variable named product
9   # print the value of product
10  product = multiply(4, 6)
11  print(product)
12
```

Lines 7-9 are comments you write. Line 10 is the call to the *multiply* function. Line 11 prints the result of the call to *multiply*. If you run the program, you will see *24* as the output.

Exercise 3: Using Copilot to Create Functions

Instructions: Write comments to guide Copilot in creating the following functions:

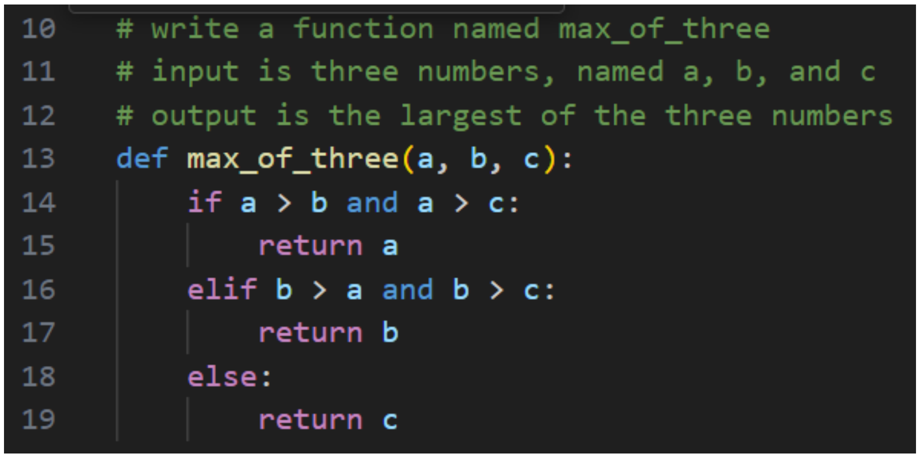
1. A function named *is_even* that takes one input, *num*, and returns *True* if the number is even and *False* if it is not. Paste a screenshot of your code below.
   Screenshot of code



```
# write a function named is_even
# input is one number, nameed num
# output is True if num is even, False if num is odd
def is_even(num):
    if num % 2 == 0:
        return True
    else:
        return False
```

Name: **Date:** **Class:**

2. A function named *max_of_three* that takes three inputs, *a, b,* and *c*, and returns the largest of the three. Paste a screenshot of your code below.
Screenshot of code

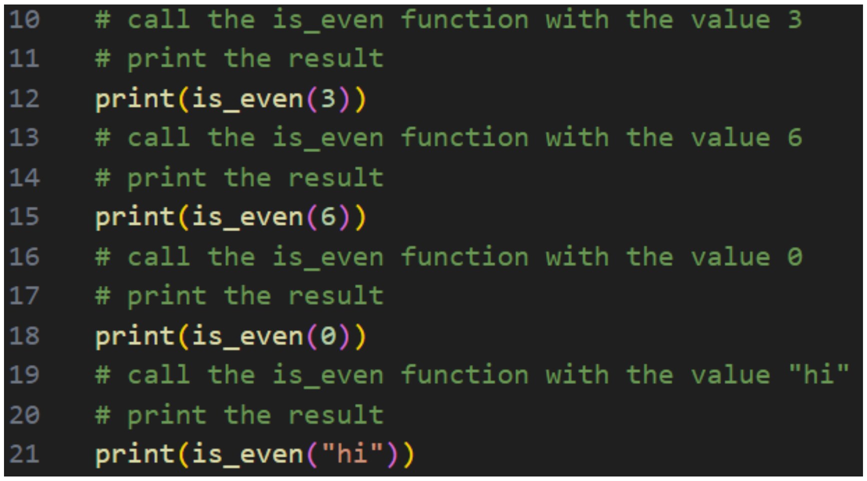


```python
# write a function named max_of_three
# input is three numbers, named a, b, and c
# output is the largest of the three numbers
def max_of_three(a, b, c):
    if a > b and a > c:
        return a
    elif b > a and b > c:
        return b
    else:
        return c
```

Exercise 4: Using Copilot to Call Functions

Instructions: Answer the following:

1. Call the *is_even* function with the following arguments:



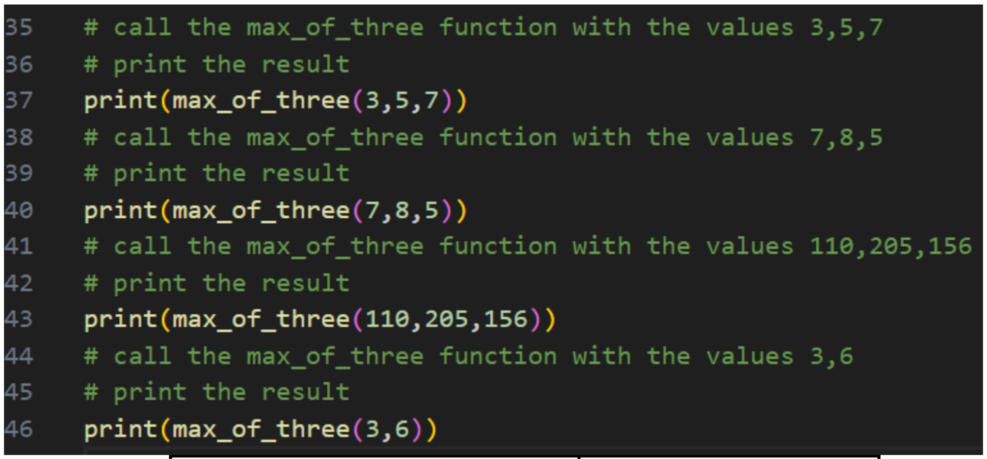
```python
# call the is_even function with the value 3
# print the result
print(is_even(3))
# call the is_even function with the value 6
# print the result
print(is_even(6))
# call the is_even function with the value 0
# print the result
print(is_even(0))
# call the is_even function with the value "hi"
# print the result
print(is_even("hi"))
```

Call to the is_even function	What is the output?
is_even(3)	False
is_even(6)	True

Name: **Date:** **Class:**

Call to the is_even function	What is the output?
is_even(0)	True
is_even("hi")	Error

2. Call the *max_of_three* function with the following arguments:



```
# call the max_of_three function with the values 3,5,7
# print the result
print(max_of_three(3,5,7))
# call the max_of_three function with the values 7,8,5
# print the result
print(max_of_three(7,8,5))
# call the max_of_three function with the values 110,205,156
# print the result
print(max_of_three(110,205,156))
# call the max_of_three function with the values 3,6
# print the result
print(max_of_three(3,6))
```

Call to the is_even function	What is the output?
max_of_three(3, 5, 7)	7
max_of_three(7, 8, 5)	8
max_of_three(110, 205, 156)	205
max_of_three(3,6)	error

Exercise 5: Reasonable Tasks for Copilot

Instructions: Discuss with your classmates and answer the following questions:

1. Why is it important to break down large problems into smaller tasks?
   Breaking down tasks makes them more manageable and easier to debug.

2. What might happen if you give Copilot a task that is too complex?
   Complex tasks may lead to errors and inefficient solutions from Copilot.

3. How can you decide what is a reasonable task for Copilot to handle?
   Reasonable tasks are clear, specific, and perform a single function.

Name: **Date:** **Class:**

Reflection
Think about a time when you had a big project or problem to solve. How did you break it down into smaller tasks? How did that help you? Write a short paragraph explaining your experience and how it relates to what you learned today. Answers will vary. Encourage students to connect their personal experiences with problem-solving and task decomposition to the concepts learned in class.