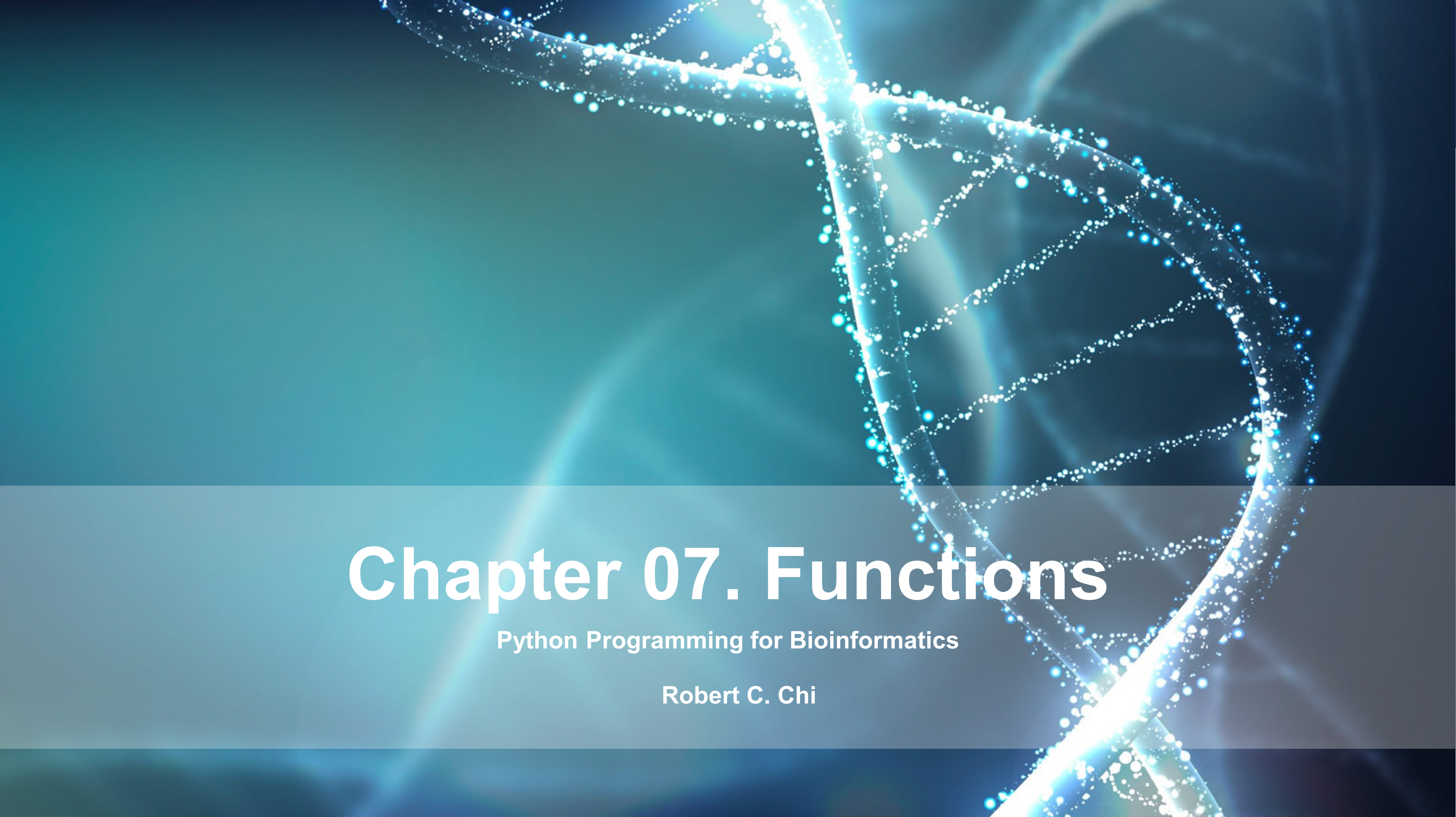


Chapter 07. Functions

Python Programming for Bioinformatics

Robert C. Chi

Agenda

- Introduction
- Definition and Calling
- Input & Return Values

INTRODUCTION

What is a "Function"?

- **A piece of code with name**

```
print(“{:*^20s}”.format(“Address Book”))
print(“1. Create a New Contact”)
print(“2. Lookup phone by name”)
print(“3. Lookup name by phone”)
print(“4. Print the whole address book”)
print(“5. Quit”)
choice = eval(input(“Your Choice: ”))
…
Other Source Codes
```

Main

```
def main_menu():
    print(“{:*^20s}”.format(“Address Book”))
    print(“1. Create a New Contact”)
    print(“2. Lookup phone by name”)
    print(“3. Lookup name by phone”)
    print(“4. Print the whole address book”)
    print(“5. Quit”)
    choice = eval(input(“Your Choice: ”))
    return choice
```

```
answer = main_menu()
print("Your Choice: {}".format(answer))
```

Main

What is a "Function"?

- Compare a “Function” between Math and Programming

Definition

$$f(x, y) = 3x + 2y$$

Calling

$$p = f(5, 4)$$
$$q = f(1, 7)$$

```
def  func(x, y):
     result = 3*x + 2*y
     return result

value1 = func(5, 4)
value2 = func(1, 7)
```

Why use "Functions"?

- **Source Code Reuse**

my_library.py

```
def main_menu():
    print("{:*^20s}".format("Address Book"))
    print("1. Create a New Contact")
    print("2. Lookup phone by name")
    print("3. Lookup name by phone")
    print("4. Print the whole address book")
    print("5. Quit")
    choice = eval(input("Your Choice: "))
    return choice
```

main1.py

```
import my_library
choice1 = my_library.main_menu()
…
```

main2.py

```
from my_library import main_menu
choice2 = main_menu()
…
```

DEFINITION AND CALLING

Define a Function

- **Syntax**

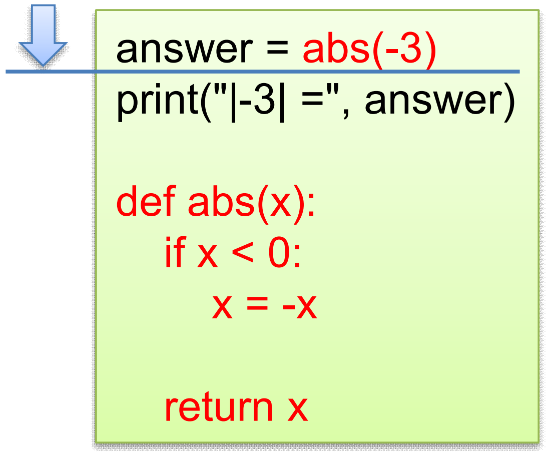
```
def <Func_Name>(param1, param2, …):
    Statement 1
    Statement 2
    …
    return <Return_Value>
```

- **Example**

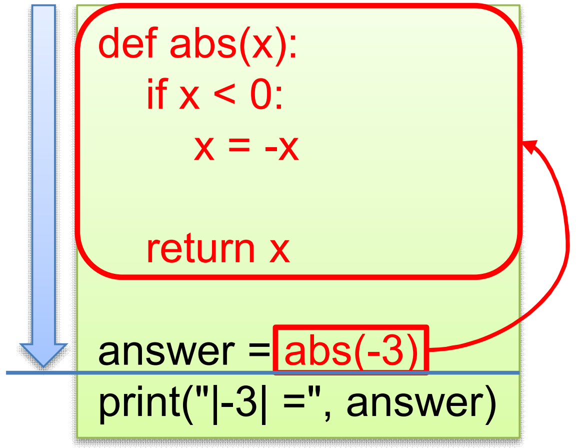
```
def abs(x):
    if (x < 0):
        x = -x

    return x
```

Where to Put Your Functions?

- **Before the Main Program:**
  - All functions should be defined before it's use



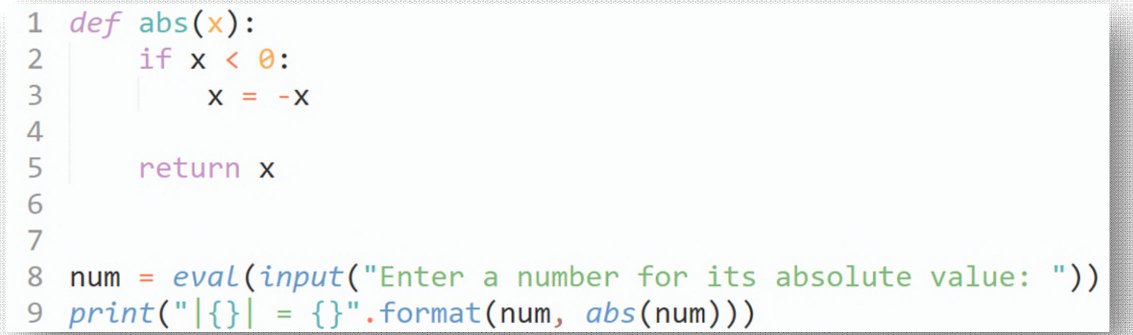
```
answer = abs(-3)
print("|-3| =", answer)

def abs(x):
    if x < 0:
        x = -x

    return x
```



```
def abs(x):
    if x < 0:
        x = -x

    return x

answer = abs(-3)
print("|-3| =", answer)
```

Practice

- **Define a Function**
  - Define a function called abs() to calculate the absolute value entered by users
  - Run the program at Google Colab.

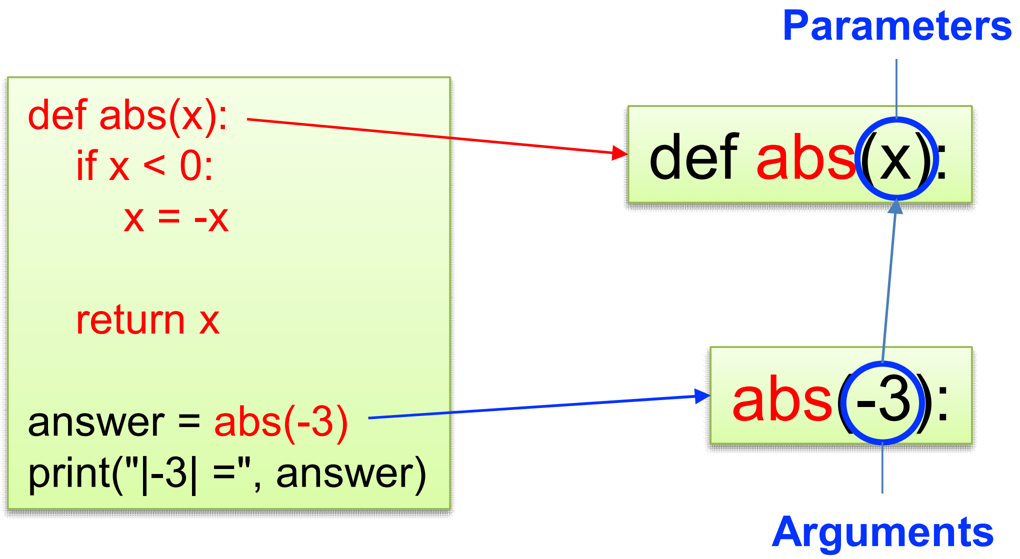


```
def abs(x):
    if x < 0:
        x = -x

    return x


num = eval(input("Enter a number for its absolute value: "))
print("|{}| = {}".format(num, abs(num)))
```

INPUT & RETURN VALUES

Parameters vs. Arguments

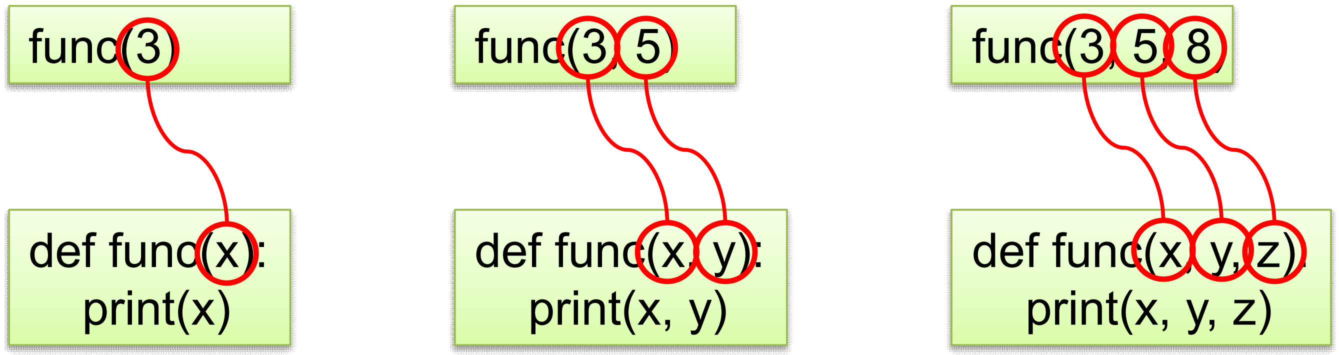
- **Parameters**: **Variables** that accept input values
- **Arguments**: Input Values **Themselves**



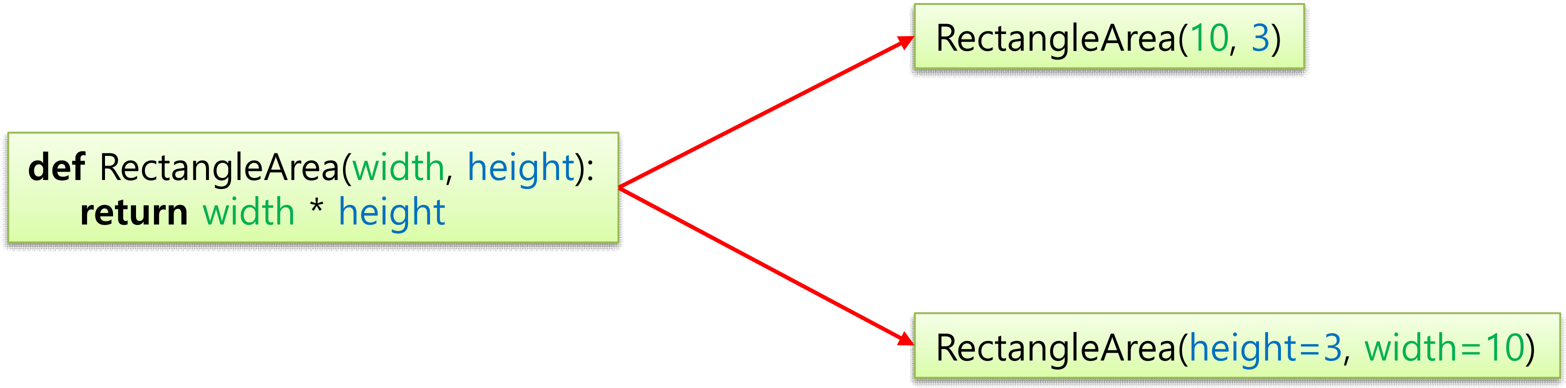
```
def abs(x):
    if x < 0:
        x = -x

    return x

answer = abs(-3)
print("|-3| =", answer)
```

Declaration and Passing of Input Values

- **By default, the number and order of input values must be the same**



Passing Arguments with Keywords



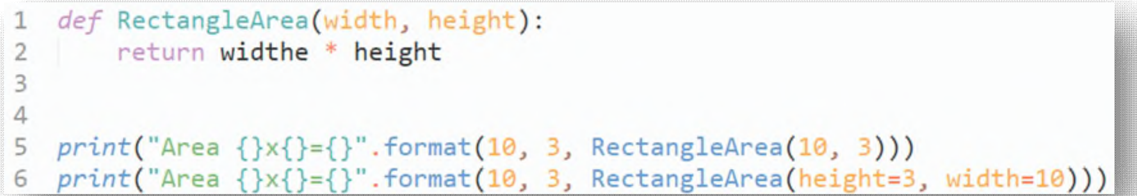
```python
def RectangleArea(width, height):
    return width * height
```

```python
RectangleArea(10, 3)
```

```python
RectangleArea(height=3, width=10)
```

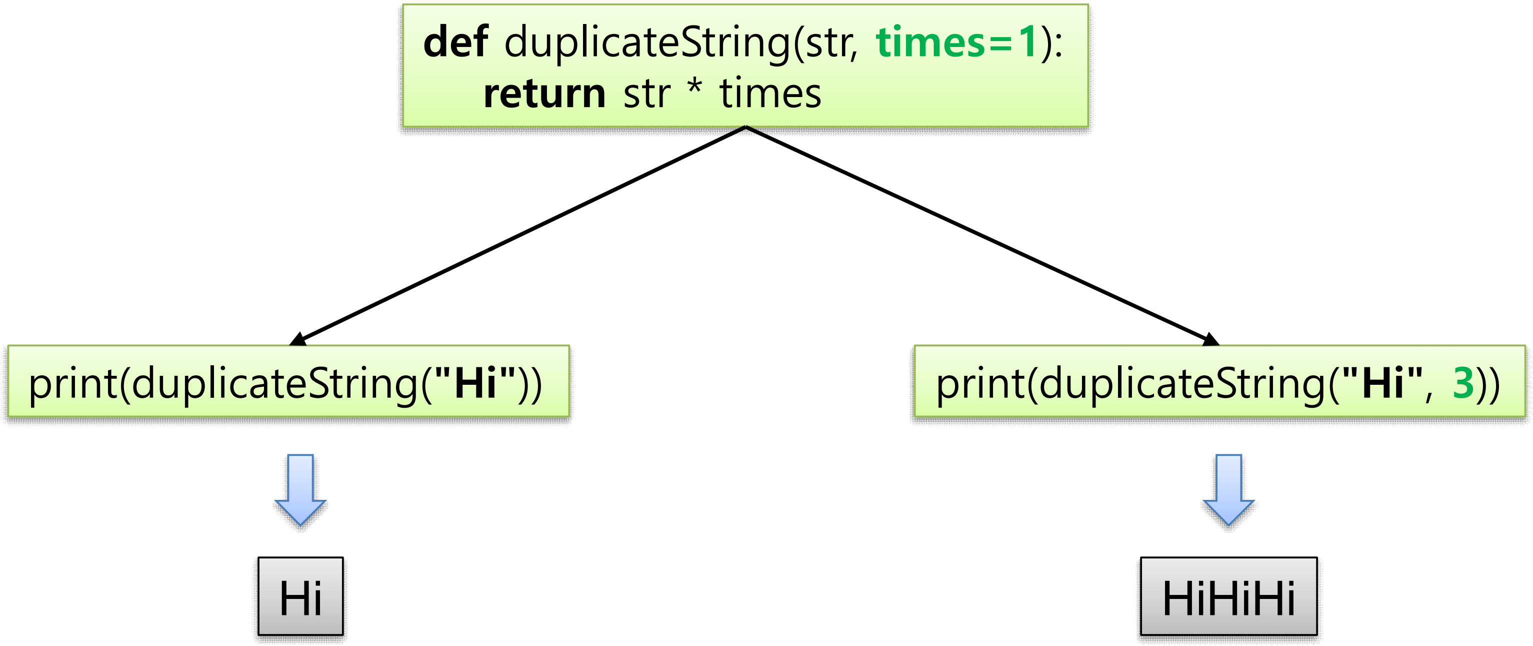
Practice

- **Passing Arguments with Keywords**
  - Define the following function:
    - def RectangleArea(width, height):
      return width * height
  - Call the function with following methods and see if the results are the same:
    - RectangleArea(10, 3)
    - RectangleArea(height=3, width=10)
  - The complete source code is shown below:



```
def RectangleArea(width, height):
    return widthe * height


print("Area {}x{}={}".format(10, 3, RectangleArea(10, 3)))
print("Area {}x{}={}".format(10, 3, RectangleArea(height=3, width=10)))
```

Omitted When Parameters have Default Values



```
def duplicateString(str, times=1):
    return str * times
```

```
print(duplicateString("Hi"))
```

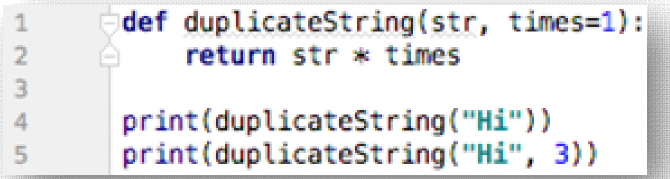
```
Hi
```

```
print(duplicateString("Hi", 3))
```

```
HiHiHi
```

Practice

- **Working with Parameters have Default Values**
  - Create the following function:
    - **def** duplicateString(str, times=1):
      return str * times
  - Call the function with following methods:
    - print(duplicateString(**“Hi”**))
    - print(duplicateString(**“Hi”**, 3))
  - The complete source code is shown below:



```
def duplicateString(str, times=1):
    return str * times

print(duplicateString("Hi"))
print(duplicateString("Hi", 3))
```

Return One Value

- **Put the Value right after the "return" keyword**

```
def square(x):
    return x * x
```

Return Multiple Values

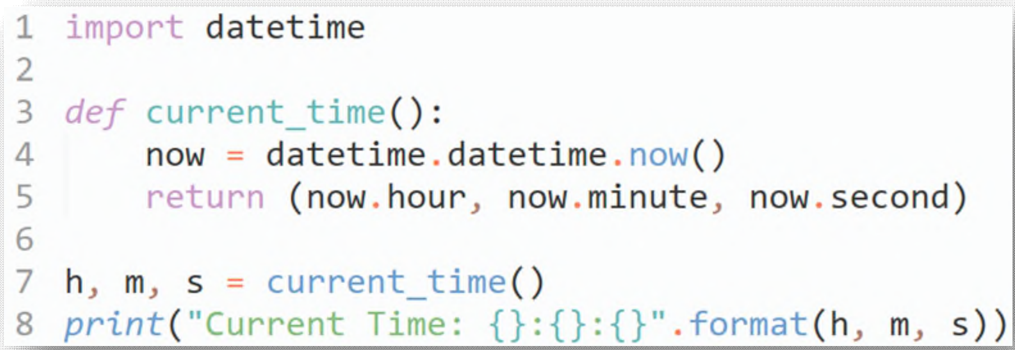
- **Return as a Compound Data Type with Un-packing**

```
import datetime

def current_time():
    now = datetime.datetime.now()
    return (now.hour, now.minute, now.second)

h, m, s = current_time()
print(" Current Time: {}:{}:{}".format(h, m, s))
```

```
Current Time: 17:52:58
```

Practice

- **Return Multiple Values**
  - Enter the following code to practice returning multiple values:



```python
import datetime

def current_time():
    now = datetime.datetime.now()
    return (now.hour, now.minute, now.second)

h, m, s = current_time()
print("Current Time: {}:{}:{}".format(h, m, s))
```