# Computer Programming in Python

Functions and Modular Programming

Examples

# Functions and Modular Programming

- Writing programs using functions
  - Understand the Requirements
    - Requirements decomposition
  - Design
    - Stepwise refinement
  - Develop
    - One logical section at a time
  - Test
    - As the program is developed (incremental testing)
    - Again, when the program is completed

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- REQUIREMENTS:
  - Write a program that obtains the price for an item, the number purchased, and computes the discount based on the total amount of the sale:
    - Sale greater than $20, discount is 10%
    - Sale greater than $40, discount is 20%
  - Two functions required:
    - One function to determine the discount
    - One function to display all of the values

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- Get a working program
  - Create the main file with the prompt for the input

```
def main():

    price_per = float(input('Enter the item price: '))

    qty = int(input('Enter the number of items: '))


    print('Price per item is: ', price_per)
    print('The number of items is: ', qty)

main()
```

```
Enter the item price: 12
Enter the number of items: 3
Price per item is:  12.0
The number of items is:  3
>>>
```

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- The first function determines the discounted total
  - To accomplish this, it will need the total for the sale
    - Total will be computed in main for the example

```
def compute_discount(total_amt):

    discount_price = total_amt

    if total_amt > 40:
        discount_price = total_amt * 0.8
    elif total_amt > 20:
        discount_price = total_amt * 0.9

    return discount_price
```

assume no discount at first

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- The function is called from main and the return value is assigned to discount

```
def main():

    price_per = float(input('Enter the item price: '))

    qty = int(input('Enter the number of items: '))

    total = price_per * qty

    discount_price = compute_discount(total)
```

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- Again, the program should be tested
  - The additional function returns the discounted total
  - A print statement is added to test the value computed

```
Enter the item price: 10
Enter the number of items: 3
Price per item is:  10.0
The number of items is:  3
Total sale:  30.0
The discounted total is:  27.0
>>>
```

*Use numbers for testing that are easy to validate*

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- The next function is required to display all of the values
  - The output should be formatted and include dollar signs as appropriate
  - First, get it working
    - The print statements from main can be moved to the function

```
def show_output(price_per, qty, total, discount_price):

    print('Price per item is: ', price_per)
    print('The number of items is: ', qty)
    print('Total sale: ', total)
    print('The discounted total is: ', discount_price )
```

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- The output function call replaces the print statements in the main program

```
def main():

    price_per = float(input('Enter the item price: '))

    qty = int(input('Enter the number of items: '))

    total = price_per * qty

    discount_price = compute_discount(total)

    show_output(price_per, qty, total, discount_price)
```

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- Again test to ensure that no bugs were introduced and then add the formatting to the output

```
def show_output(price_per, qty, total, discount_price):

    print('Price per item is: $', format(price_per, '.2f'))
    print('The number of items is: ', qty)
    print('Total sale: $', format(total, '.2f'))
    print('The discounted total is: $', format(discount_price, '.2f'))
```

```
Enter the item price: 10
Enter the number of items: 3
Price per item is: $ 10.00
The number of items is:  3
Total sale: $ 30.00
The discounted total is: $ 27.00
>>>
```

Chris Simber 2021 - All Rights Reserved

# Another example

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- Grade Averaging Example:
  - REQUIREMENTS:
    - Write a program that obtains five (5) grades from the user, computes the average, and assigns a letter grade to the result
    - **Three (3) functions**
      - get_input()
      - get_average()
      - assign_letter()

*Name functions what they do…*

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- Grade Averaging Example:
  - get_input()
    - Obtains the five grades from the user and returns the total
  - get_average()
    - Determines and returns the average based on the total
  - assign_letter()
    - Assigns and returns the letter grade based on the average

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- Grade Averaging Example:
  - get_input()
    - Obtains the five grades from the user and returns the total

```
def get_input():

    total = 0

    for num in range(5):
        grade = int(input('Enter grade #' + str(num + 1) + ': '))
        total = total + grade

    return(total)
```

parentheses optional

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- Grade Averaging Example:
  - Create main and test the function

```
def main():

    total = get_input()
```

```
Enter grade #1: 2
Enter grade #2: 2
Enter grade #3: 2
Enter grade #4: 2
Enter grade #5: 2
```

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- Grade Averaging Example:
  - get_average()
    - Determines and returns the average based on the total

```
def get_average(total_grades):

    average = 0

    if total_grades > 0:
        average = total_grades/5

    return average
```

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- Grade Averaging Example:
  - Modify main to call the function and print the result

```
def main():

    total = get_input()
    average = get_average(total)
    print('total is: ', total, ' and average is: ', average)
```

```
Enter grade #1: 60
Enter grade #2: 70
Enter grade #3: 80
Enter grade #4: 90
Enter grade #5: 100
total is:  400  and average is:  80.0
>>>
```

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- Grade Averaging Example:
  - assign_letter()
    - Assigns and returns the letter grade from the average

```
def get_letter(average_grade):

    letter = 'F'

    if average_grade >= 90:
        letter = 'A'
    elif average_grade >= 80:
        letter = 'B'
    elif average_grade >= 70:
        letter = 'C'
    elif average_grade >= 60:
        letter = 'D'

    return (letter)
```

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- Grade Averaging Example:
  - Test the program

```
Enter grade #1: 60
Enter grade #2: 70
Enter grade #3: 80
Enter grade #4: 90
Enter grade #5: 100
total is:  400
The average is 80.0
The letter grade is  B
>>>
```

```
def main():

    total = get_input()
    average = get_average(total)
    letter = get_letter(average)

    print('total is: ', total)
    print('The average is', average)
    print('The letter grade is ', letter)
```

Chris Simber 2021 - All Rights Reserved

# Modular Programming

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- Grade Averaging Example:
  - Modularizing would place the functions in a separate file
  - The file would be imported into the main program file
  - The function calls would include the name of the other file before the name of the function

```
import grades

def main():

    total = grades.get_input()
```

name of file containing functions

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- Grade Averaging Example:
  - Modularized main

```
import grades

def main():

    total = grades.get_input()
    average = grades.get_average(total)
    letter = grades.get_letter(average)

    print('total is: ', total)
    print('The average is', average)
    print('The letter grade is ', letter)


main()
```

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- Module containing the functions

grades.py - E:/Python_Book_Programs/CH_6/grades.py (3.9.0)

File Edit Format Run Options Window Help

```python
# Obtains five grades as itegers from the user
def get_input():

    total = 0

    for num in range(5):
        grade = int(input('Enter grade #' + str(num + 1) + ': '))
        total = total + grade

    return (total)



# Computes the average from the grade total
def get_average(total_grades):

    average = 0

    if total_grades > 0:
        average = total_grades/5

    return (average)



# Assigns a letter grade based on the average
def get_letter(average_grade):

    letter = 'F'

    if average_grade >= 90:
        letter = 'A'
    elif average_grade >= 80:
        letter = 'B'
    elif average_grade >= 70:
        letter = 'C'
    elif average_grade >= 60:
        letter = 'D'

    return (letter)
```

Ln: 27 Col: 0

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- Using two (2) files - Steps:
  - Create the main module (file) and name it
  - Create the second module (file) and name it
  - The second module is imported into the main module
  - Write the functions in the second module
  - Call the functions in the second module using the module name and the dot operator from main

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- Using two (2) files simply separates the functions into another module (file)
  - Rectangle Program Example:
    - Write a program that uses separate functions to obtain the side lengths of the rectangle from the user, and to compute the area, perimeter, and diagonal of the rectangle
    - Validate the input, and use a function to display the results
    - Locate the functions in a separate module

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- Create the main module
- Create the file for the functions
- Import the function file into the main file
  - Then, step-by-step design and programming, but…

```
def main():

    side1 = get_side_length ???????
    side2 =
```

  - How to do we use a function to get two side lengths?

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- There are several ways to obtain the two side lengths from the user in a function
  - For this example, the function will be called twice
  - An extra parameter passed to the function can clarify to the user what is requested

```
import rectangle

def main():

    side1 = rectangle.get_side_length('first')
    side2 = rectangle.get_side_length('second')
```

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- The argument passed to the function is used in the prompt to the user
- The loop validates the input

```
side1 = rectangle.get_side_length('first')
side2 = rectangle.get_side_length('second')
```

```
def get_side_length(number):

    length = 0

    while length <= 0:
        length = float(input('Enter the ' + number + ' side length: '))

    return length
```

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- Using two (2) files Programming the Rectangle
  - Consider the design of the main program

```
# get the area from a function in the other file

# get the perimeter from a function in the other file

# get the diagonal from a function in the other file

# show the output from a function in the other file
```

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- The functions are added to the function module
- Testing should be completed for each function

```
def get_area(sideOne, sideTwo):
    area = sideOne * sideTwo
    return area


def get_perimeter(sideOne, sideTwo):
    perimeter = sideOne * 2 + sideTwo * 2
    return perimeter
```

```
Enter the first side length: 3
Enter the second side length: 4
The area is:  12.0
The perimeter is:  14.0
>>>
```

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- The diagonal and output functions are next
  - The diagonal function will use sqrt() which requires the math module

```
def get_diagonal(sideOne, sideTwo):
    squareSum = sideOne * sideOne + sideTwo * sideTwo
    diag = math.sqrt(squareSum)
    return diag
```

**import math**

```
def show_output(area, perimeter, diagonal):
    print('\n  The area is: ' + format(area, '.1f') +
          '\n  The perimeter is: ' + format(perimeter, '.1f')\
          + '\n  The diagonal is: ' + format(diagonal, '.1f'))
```

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- Then, test again
  - A 3, 4, 5 triangle works well
    - Since ‘a’ squared + ‘b’ squared = ‘c’ squared

```
Enter the first side length: 3
Enter the second side length: 4

  The area is: 12.0
  The perimeter is: 14.0
  The diagonal is: 5.0
>>>
```

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- Complete Main file

```
import rectangle

def main():

    side1 = rectangle.get_side_length('first')
    side2 = rectangle.get_side_length('second')

    area = rectangle.get_area(side1, side2)

    perimeter = rectangle.get_perimeter(side1, side2)

    diagonal = rectangle.get_diagonal(side1, side2)

    rectangle.show_output(area, perimeter, diagonal)


main()
```

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- Rectangle.py

```
import math

def get_side_length(number):

    length = 0

    while length <= 0:
        length = float(input('Enter the ' + number + ' side length: '))

    return length



def get_area(sideOne, sideTwo):
    area = sideOne * sideTwo
    return area


def get_perimeter(sideOne, sideTwo):
    perimeter = sideOne * 2 + sideTwo * 2
    return perimeter
```

Chris Simber 2021 - All Rights Reserved

# Functions and Modular Programming

- Rectangle.py

```
def get_diagonal(sideOne, sideTwo):
    squareSum = sideOne * sideOne + sideTwo * sideTwo
    diag = math.sqrt(squareSum)
    return diag


def show_output(area, perimeter, diagonal):
    print('\n  The area is: ' + format(area, '.1f') +
          '\n  The perimeter is: ' + format(perimeter, '.1f')\
          + '\n  The diagonal is: ' + format(diagonal, '.1f'))
```

Chris Simber 2021 - All Rights Reserved

# *Functions and Modular Programming*

Chris Simber 2021 - All Rights Reserved