Practice with Functions

Learning Objectives:

- Be able to define functions
- Be able to call functions
- Be able to pass arguments into functions
- Be able to return values from functions
- Be able to comment functions so they appear in the help function

Prerequisites: Before starting this lab, you should be able to:

- Compose programs using good programming habits, that have headers, and that make use of comments, variables, basic arithmetic, data types, and basic I/O statements
- Use the divmod() function

Task 1: Determining BCS Scores (pair task)

A college team’s BCS football ranking is comprised of three elements: 1) results from the Harris Poll, 2) results from the Coaches Poll, and 3) results from computer rankings. Each of these counts toward one-third of the final BCS ranking.

A team’s ranking in the Harris Poll is divided by 2,850, which is the maximum number of points any team can receive if all 114 voting members rank the same team as number 1.

A team’s ranking in the Coaches Poll is divided by 1,475, which is the maximum number of points any team can receive if all 59 voting members rank the same team as number 1.

The computer rankings are calculated by some magical formula that we won’t concern ourselves over. We do know, however, that it falls between 0 and 1.

Write a program to calculate BCS scores for college football teams. Prompt a user for the team’s Harris Poll ranking (an integer between 1 and 2,850), for the team’s Coaches Poll ranking (an integer between 1 and 1,475), and the team’s computer ranking (a float between 0 and 1). Your program should include:

- A function calc_harris_score() that calculates the Harris Poll score from the Harris Poll ranking
- A function calc_coaches_score() that calculates the Coaches Poll score from the Coaches Poll ranking
A function `calc_bcs_score()` that calculates the BCS score from the Harris Poll score, the Coaches Poll score, and the computer ranking.

The results should be displayed for the Harris Poll score, the Coaches Poll score, the computer ranking, and the BCS score. Examples follow:

1. Enter the team’s Harris Poll ranking (1-2850): 1730
2. Enter the team’s Coaches Poll ranking (1-1475): 1000
3. Enter the team’s computer ranking (0-1): 0.65

Harris Poll score: 0.6070175438596491
Coaches Poll score: 0.6779661016949152
Computer ranking: 0.65
Resulting BCS score: 0.6449945485181882

4. Enter the team’s Harris Poll ranking (1-2850): 2000
5. Enter the team’s Coaches Poll ranking (1-1475): 1300
6. Enter the team’s computer ranking (0-1): 0.8

Harris Poll score: 0.7017543859649122
Coaches Poll score: 0.8813559322033898
Computer ranking: 0.8
Resulting BCS score: 0.7943701060561006

After your program and functions are working properly, show them to your TA to get credit.

Task 2: The Doomsday Algorithm Once More (pair task)

A year’s Doomsday is given by the equation:

\[
Doomsday = (((yr \div 12) + (yr \mod 12) + (yr \mod 12) \div 4) \mod 7 + anchor) \mod 7
\]  

(1)

where `yr` is the last two digits of the year and the anchor day is an integer between 0 and 7 with 0 corresponding to Sunday and 7 to Saturday.

Write a program to calculate a year’s Doomsday. Prompt a user for the last two digits of the year and for the anchor day as an integer. Print out the Doomsday. Include the following:

- A function `calc_doom()` that calculates the doomsday that takes two parameters `yr` and `anchor` and returns the doomsday
- A docstring in the function (i.e., right under the “def” line of the function surrounded by three single or double quotes) that explains the purpose of the function

Anchor days for 2000-2099 are Tuesday (=2) and for 1900-1999 Wednesday (=3). Example results are shown below.
Enter the last two digits of the year: 05
Enter the anchor day as an integer [0=Sunday, 7=Saturday]: 2
Doomsday = 1

Enter the last two digits of the year: 17
Enter the anchor day as an integer [0=Sunday, 7=Saturday]: 2
Doomsday = 2

Run the program for your TA to get credit.

Task 3: Body Mass Index and Metric Dimensions (pair task)

The body mass index (BMI) is given by the formula

\[
\frac{\text{weight}}{(\text{height})^2}
\]

where \text{weight} is weight (or, more precisely, mass) in kilograms and \text{height} is height in meters. Write a program that does the following:

- Prompts a user for his or her weight in pounds and height in inches
- Declares global variables \text{kg\ per\ lb} = 0.45359237 and \text{m\ per\ inch} = 0.0254
- Calls a function \text{convert\ height()} with parameter \text{height} that converts height to meters
- Calls a function \text{convert\ weight()} with parameter \text{weight} that converts weight to kilograms (i.e., mass)
- Calls a function \text{calc\ bmi()} with parameters \text{height\ m} and \text{mass}
- Prints the results as given in the examples below

Example:
Enter weight [pounds]: 160
Enter height [inches]: 72

\[
\begin{align*}
\text{BMI} & = 21.6996783839 \\
\text{Mass} & = 72.5747792 \text{ [kilograms]} \\
\text{Height} & = 1.8288 \text{ [meters]}
\end{align*}
\]

Another example (this time with float inputs):
Enter weight [pounds]: 130.5
Enter height [inches]: 69.5

BMI = 18.9949961478
Mass = 59.193804285 [kilograms]
Height = 1.7653 [meters]

After your program is working properly, show it to your TA and demonstrate that it does what it should.

---

**Task 4: Student Loan Calculator Again (pair task)**

Next you’re going to modify your Student Loan Repayment Calculator program to use functions. As a reminder, the equation for a monthly payment is given by:

$$ p = \frac{P(i/12)}{1 - (1 + i/12)^{-n}} $$

where $p$ is the monthly payment, $P$ is the loan amount, $i$ is the annual percent interest rate (must be a fraction in the program), and $n$ is the number of monthly payments. Write a program to calculate the monthly amount you owe, the total amount you’ll end up paying, and the total amount of interest you’ll have paid. The program must include the following:

- Global variables for the interest rate (a float), number of monthly payments (an integer), and amount of loan principal ($P$, a float)
- A function `calc_monthly_pymt()` that calculates your monthly payment
- A function `display_money()` with a parameter `dollar_amount` that’s used to display each of the different amounts of money in a nice format.

Hint: For `display_money()`, multiply the dollar amount by 100 to get the total cents, use `divmod()` twice, and return a string, e.g., `str(dollars) + '.' + str(dimes) + str(pennies)`.

Here’s an example of my output:

Enter the amount you owe [no commas]: 30000
Enter the interest rate [%]: 6.8
Enter the number of years you want to spend to pay back your loan: 10

Your monthly payment is $345.24.
The total amount you ended up paying is $41428.92.
The total amount of interest you paid is $11428.92.

Demonstrate your program to your TA to get credit.
**Task 5: Making Change Again (pair task)**

Again write a program that prompts a user for the total cost of a purchase and the cash payment made using a U.S. currency bill. This time, however, your program needs to include a function `make_change` with parameter `change` that figures out how many dollars, quarters, dimes, nickels, and pennies the customer should be given. Your program should display the amount of change due to the customer as well as the number of dollars, quarters, dimes, nickels, and pennies.

Hint: Again, multiply the amount of change by 100 and use the `divmod()` function four times. **IMPORTANTLY**, when you call `make_change()`, use the following command: `dollars, quarters, dimes, nickels, pennies = make_change(change)` and in the function use `return dollars, quarters, dimes, nickels, pennies`. These are called tuples, and we’ll study them later.

Your program display should be similar to the following:

```
1 Enter the total cost: 14.32
2 Enter the cash payment: 20
3
4 Change: $5.68
5   Dollars:  5
6   Quarters: 2
7   Dimes:  1
8   Nickels: 1
9   Pennies: 3
```

Run your program and show it to your TA to get credit.

---

**Task 6: From Forward to Reverse (pair task)**

Write a program that prompts the user for a four-digit, positive integer and then calls a function called `reverse_digits()` to print the digits in reverse order on one line with no spaces between digits. It’s not actually necessary to use the `end` parameter in the `print()` function, but you’ll have to use either it or the `sep` parameter.

Your output should look similar to this:

```
1 Enter a four-digit, positive integer: 1234
2 1234

3 If the input has trailing zeros, the reversed number will look a bit odd, but it’s okay.
4 Enter a four-digit integer: 1200
5 1001
```

After your program and function are running correctly, show them to your TA to get credit.