2.4 Writing Our Own Functions

Motivation for User-Defined Functions

Up until now, we've only called functions that Python has provided for us. Some examples of the functions we've used are: print, input, int, float, sqrt, round and randint. Python and nearly all programming languages allow users to define their own functions.
Simply put, a function is a mini-program that completes a specified task. For example, the sqrt function takes its input value and returns its square root. The randint function takes in a low bound and a high bound and selects a random integer within the specified range. The ceil function finds the smallest integer greater than or equal to its input value and returns it. We can define functions to do any sort of task that we would like or to make any sort of calculation we want. Once we define a function, we can use it over and over again.

Function Example

Consider a game where two people both roll a pair of regular dice and the first person wins if their roll sums to a higher number than the other person. (Thus, if both people roll the same value, then the second person wins.) In this program, a task we need to repeat is rolling a pair of dice. First the code will be shown in its entirety and then an explanation will be given for how it works:

```
import random

def main():

    random.seed()

    # Roll both pairs of dice.
    score1 = rollPairDice()
    score2 = rollPairDice()

    print("Player 1 rolled a",score1,"and player 2 rolled a",score2)

    # Print out the winner.
    if score1 > score2:
        print("Player 1, you win!")
    else:
        print("Player 2, you win!")

def rollPairDice():
    return random.randint(1,6) + random.randint(1,6)

main()
```

We first define each of our functions. When we define our functions, they don't actually get interpreted. Rather, we are defining them so that when we call them later, the interpreter knows what the function calls mean. Thus, before we ever call main, both the function main and rollPairDice are defined.

As we interpret main, we run across the call to rollPairDice. We then start executing the steps of this function. This function has a single statement, which, itself, contains a couple function calls. The first randint call will generate a random integer in between 1 and 6 and return it. The second randint call will do the same thing. Note that this second call may very well return a different number than the first call. The function adds these two values and then returns this sum. The return statement is a special statement designated for functions. The return statement indicates that the function is ending and that it is returning a value to the function that called it.

In this case, the rollPairDice function returns a value to main. This value is then stored in the variable score1. We repeat this process, calling the rollPairDice function a second time. Once it finishes, the value it returns gets stored in score2. Note that any time a function returns a value, the function that calls it should store that value in a variable or use it in some way.

Once these lines are done, the program proceeds as we'd expect, comparing the two variables and printing out the appropriate message.

Function Example with Different-Sided Dice

Though most games with dice have their users roll a pair of 6-sided dice, not all games are this way. We can make our function more general by allowing the user to determine the number of sides on each of the two dice. In order for this to occur, our function needs some information. It needs to know how many sides are on each die. This information is given to a function in its parameter list. A parameter list is a list of information the function needs to do its task. Here is our modified function definition:

```
def rollPairDice(numSides):
    return random.randint(1,numSides) + random.randint(1,numSides)
```

Now, in order to call this function, we must put a number in the parentheses in between rollPairDice. Now, let's look at the modified program in full that accomplishes the same exact task as the previous program:

```
import random

def main():
    random.seed()

    # Roll both pairs of dice.
    score1 = rollPairDice(6)
    score2 = rollPairDice(6)

    print("Player 1 rolled a",score1,"and player 2 rolled a",score2)

    # Print out the winner.
    if score1 > score2:
        print("Player 1, you win!")
    else:
        print("Player 2, you win!")
```

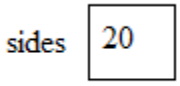
```
def rollPairDice(numSides):
    return random.randint(1,numSides) + random.randint(1,numSides)

main()
```

Extending Our Program

Now that we have a more general function, we can use it to write a program that allows users to play this game with a pair of Dice with any number of sides:

```
import random

def main():

    random.seed()

    # Get the number of sides on each die.
    sides = int(input("How many sides are on each die?\n"))

    # Roll both pairs of dice.
    score1 = rollPairDice(sides)
    score2 = rollPairDice(sides)

    print("Player 1 rolled a",score1,"and player 2 rolled a",score2)

    # Print out the winner.
    if score1 > score2:
        print("Player 1, you win!")
    else:
        print("Player 2, you win!")

# Returns the sum of rolling two dice labeled 1 to numSides.
def rollPairDice(numSides):
    return random.randint(1,numSides) + random.randint(1,numSides)

main()
```

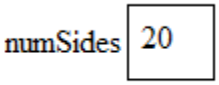
In this example, we can see the mechanics of functions working. After the second line of code in main, the picture of memory in main is as follows, assuming that the user enters 20 for the number of sides on a die:



Now, consider what occurs when we make the following function call:

```
score1 = rollPairDice(sides)
```

Before the rollPairDice function runs, the first task is assigning each parameter to the appropriate value. Each function gets its own memory for each time it runs. Thus, our picture for the function is as follows:



Essentially, what has happened is that the value of sides in main (20), has been copied into the box for numSides in the function rollPairDice. From here, we run the function, which itself calls the randint function, which will return a random integer in between 1 and 20, inclusive. This function gets called a second time and the second random integer returned is added to the first, and then this sum is returned to main and stored in score1.

This whole process is repeated again. For our purposes, we call the rollPairDice function a second time with the same parameter (sides), but there's no reason why we couldn't call it with a different parameter.