Instead of repeating several lines of code every time you want to perform a particular task, use a function that will allow you to write a block of Python code once and then use it many times. This can help reduce the overall length of your programs, as well as give you a single place to change, test, troubleshoot, and document any given task. Ultimately, this makes your application much easier to maintain in the long run.

To create a function, use the **def** keyword followed by the name of the function. Always follow the function name with a set of parentheses. If your function accepts parameters you may include the names of those parameters within the parentheses, separating them with commas. Finally, conclude the function definition line with a colon. The code block that follows the function definition will be executed any time the function is called. The pattern is **def function_name():**.

Creating a simple function:

```
def say_hello():       # defining a function using def keyword
        print("Hello Python Programmers")

>>> say_hello()      # displaying the contents of a function or printing a function
Hello Python Programmers
```

Creating a function with a parameter

```
def say_hello(name):
        print("Welcome {}".format(name))

>>> say_hello("Vamshi")
Welcome Vamshi
```

Creating a function with optional parameter

If you'd like to make the parameter optional, set a default value for it by using the equals sign. The pattern is **def function_name(parameter_name = default_value):**

```
def say_hello(name = 'Vamshi'):
        print("Hello! {}".format(name))

>>> say_hello()
Hello! Vamshi
>>> say_hello('all')
Hello! all
```

Creating a function with multiple parameters [named parameters]

Keep in mind that functions are capable of accepting multiple parameters. All you need to do is include them within the parentheses of the function definition, and separate them with a

comma. When you are calling the function, always supply the arguments and separate them with commas as well.

Method 1:

```
def say_hello(first, last):
        print("Hello {} {}".format(first, last))

>>> say_hello("Vamshi", "G")
Hello Vamshi G
```

Method 2:

```
def say_hello(first, last):
        print("Welcome {} {}".format(first, last))

>>> say_hello(first = 'Vamshi', last = 'G')
Welcome Vamshi G
>>> say_hello(first = 'Aditya', last = 'V')
Welcome Aditya V
```

Method 3:

It is also possible to combine required and optional parameters as in the following example. If you use both required and optional parameters, the required parameters should come first.

```
def say_hello(first, last = 'G'):
        print("Hello! {} {}".format(first, last))

>>> say_hello('Vamshi')
Hello! Vamshi G
>>> say_hello('Vamshi', 'Gulaigiri')
Hello! Vamshi Gulaigiri
```

Creating a function with a return statement:

Not only are functions capable of performing a task, they can also return data by using the **return** statement. This statement makes it possible for you to return any data type that you require. Once the **return** statement is called, no further code in the function will be executed. The following code is a function that returns a string.

```
def even_or_odd(number):
        """Determine if a number is even or odd."""
```

```
        if number % 2 == 0:
                return 'Even'
        else:
                return 'Odd'


>>> even_or_odd(5)
'Odd'
>>> even_or_odd(2)
'Even'
```

Example code:

```
def is_odd(number):
        if number % 2 == 1:
                return 'True'
        else:
                return 'False'

>>> is_odd(9)
'True'
>>> is_odd(4)
'False'
```

Calling a function within another function

It is entirely possible for you to create functions that call other functions. The following listing is an example

```
def get_name():
        """Get and return a name"""
        name = raw_input("What's your name?")
        return name

>>> def say_name(name):
        """Say a name"""
        print("So your name is {}".format(name))

>>> def get_and_say_name():
        """Get and display name"""
        name = get_name()
        say_name(name)
        print("Welcome! {}".format(name))
```

```
>>> # displaying the output
get_and_say_name()

What's your name? Vamshi
So your name is  Vamshi
Welcome!  Vamshi
```