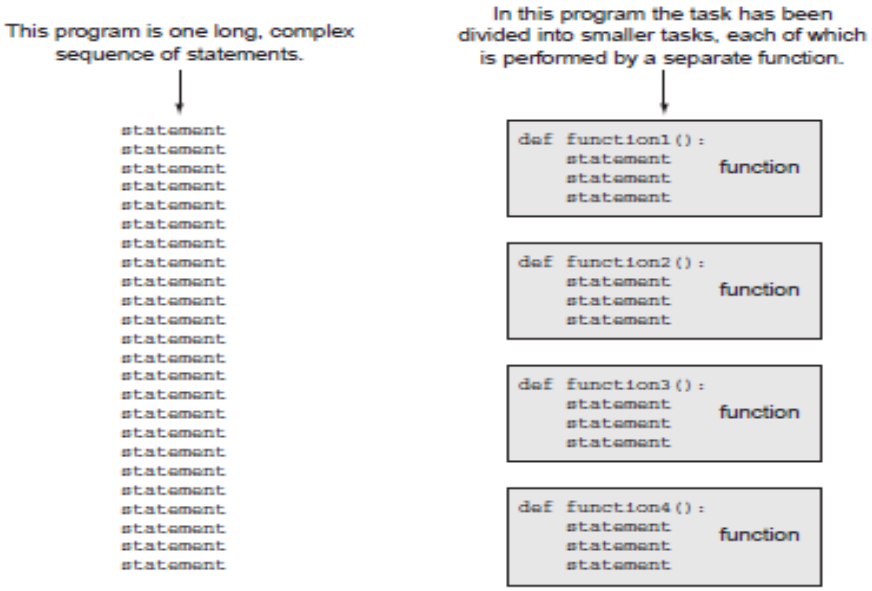
Functions

1. **Functions** A function is a group of statements that exist within a program for the purpose of performing a specific task.



2. **Void and Value Returning Functions**

When you call a ***void function,*** it simply executes the statements it contains and then terminates. When you call a *value-returning function*, it executes the statements that it contains, and then it returns a value back to the statement that called it.

Eg. Input function is an example of value returning function.

3. **Defining a function and function call**

```
def function_name():
    statement
    statement
    etc.
```

These statements cause the `message` function to be created.

```
# This program demonstrates a function.
# First, we define a function named message.
def message():
    print('I am Arthur,')
    print('King of the Britons.')

# Call the message function.
message()
```

This statement calls the `message` function, causing it to execute.

4. **Calling the nested function**

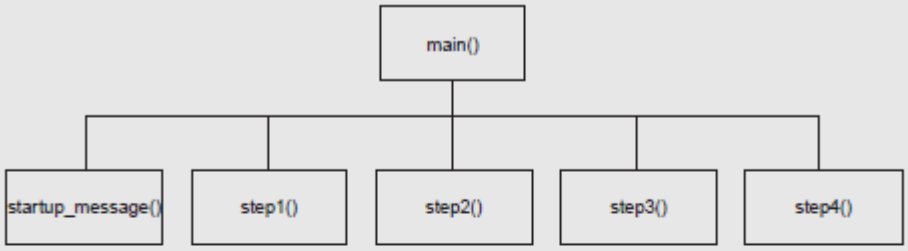
The interpreter jumps to the `message` function and begins executing the statements in its block.

```
# This program has two functions. First we
# define the main function.
def main():
    print('I have a message for you.')
    message()
    print('Goodbye!')

# Next we define the message function.
def message():
    print('I am Arthur,')
    print('King of the Britons.')

# Call the main function.
main()
```

5. **Indentation in Python :** Block must be Indented**,** Python Programmer primarily use four spaces to indent the lines in a block. Blank lines are ignored



As you can see from the hierarchy chart, the `main` function will call several other functions. Here are summaries of those functions:

- `startup_message`—This function will display the starting message that tells the technician what the program does.
- `step1`—This function will display the instructions for step 1.
- `step2`—This function will display the instructions for step 2.
- `step3`—This function will display the instructions for step 3.
- `step4`—This function will display the instructions for step 4.

Between calls to these functions, the `main` function will instruct the user to press a key to see the next step in the instructions. Program 5-3 shows the code for the program.

6. **Scope of variables**

A variable is visible only to statements in the variable's scope. A local variable's scope is the function in which the variable is created.

```
def main():
    # Call the texas function.
    texas()
    # Call the california function.
    california()

# Definition of the texas function. It creates
# a local variable named birds.
def texas():
    birds = 5000
    print('texas has', birds, 'birds.')

# Definition of the california function. It also
# creates a local variable named birds.
def california():
    birds = 8000
    print('california has', birds, 'birds.')

# Call the main function.
main()
```

7. **Passing arguments to functions**

An argument is any piece of data that is passed into a function when the function is called. A parameter is a variable that receives an argument that is passed into a function

A parameter variable, often simply called a parameter, is a special variable that is assigned the value of an argument when a function is called. Here is an example of a function that has a parameter variable:

```
def show_double(number):
        result =number *2
        print(result)
```

```
# This program demonstrates an argument being
# passed to a function.

def main():
    value = 5
    show_double(value)

# The show_double function accepts an argument
# and displays double its value.
def show_double(number):
    result = number * 2
    print(result)

# Call the main function.
main()
```

8. **Passing Multiple Arguments**

```
>>> def main():
        print(' the Sum of 12 and 45 is')
        show_sum(12,25)
>>> def show_sum(num1,num2):
        result= num1+num2
        print(result)
 >>> main()
```

9. **Global variable and Local Variable**

```
>>> number=0
>>> def main():
        global number
        number=4
        show_number()
>>> def show_number():
        print('the number',number)
```

10. **Generating random numbers**

```
Import random
>>> for i in range(1,20):
        no=random.randint(1,20)
        print(no)
```

Problem :Dr. Kimura teaches an introductory statistics class and has asked you to write a program that he can use in class to simulate the rolling of dice. The program should randomly generate two numbers in the range of 1 through 6 and display them. In your interview with Dr. Kimura, you learn that he would like to use the program to simulate several rolls of the dice, one after the other. Here is the pseudocode for the program:

While the user wants to roll the dice:
    Display a random number in the range of 1 through 6
    Display another random number in the range of 1 through 6
    Ask the user if he or she wants to roll the dice again

11. **Random seed function**

The formula that generates random numbers has to be initialized with a value known as a seed value.

Standard Library functions and import statements

12. **Writing own value returning function**

A value-returning function has a return statement that returns a value back to the part of the program that called it.

```
>>> def sum(n1,n2):
        result=n1+n2
        return result
```

```
>>> def sum(n1,n2):
        result=n1+n2
```

13. Return strings

```
def get_name():
    # Get the user's name.
    name = input('Enter your name: ')
    # Return the name.
    return name
```

14. Return Boolean values

```
def is_even(number):
    # Determine whether number is even. If it is,
    # set status to true. Otherwise, set status
    # to false.
    if (number % 2) == 0:
        status = True
    else:
        status = False
    # Return the value of the status variable.
    return status
```

15. Returning Multiple values

16. Math Module

The Python standard library's math module contains numerous functions that can be used in mathematical calculations

Import math

math.sqrt(number)

math.hypot(a,b)

`math` Module Function	Description
`acos(x)`	Returns the arc cosine of `x`, in radians.
`asin(x)`	Returns the arc sine of `x`, in radians.
`atan(x)`	Returns the arc tangent of `x`, in radians.
`ceil(x)`	Returns the smallest integer that is greater than or equal to `x`.
`cos(x)`	Returns the cosine of `x` in radians.
`degrees(x)`	Assuming `x` is an angle in radians, the function returns the angle converted to degrees.
`exp(x)`	Returns e^x
`floor(x)`	Returns the largest integer that is less than or equal to `x`.
`hypot(x, y)`	Returns the length of a hypotenuse that extends from (0, 0) to (`x`, `y`).
`log(x)`	Returns the natural logarithm of `x`.
`log10(x)`	Returns the base-10 logarithm of `x`.
`radians(x)`	Assuming `x` is an angle in degrees, the function returns the angle converted to radians.
`sin(x)`	Returns the sine of `x` in radians.
`sqrt(x)`	Returns the square root of `x`.
`tan(x)`	Returns the tangent of `x` in radians.

17. Creating the module

```
>>> import circle
>>> print('area',circle.area(3))
area 28.274333882308138
```